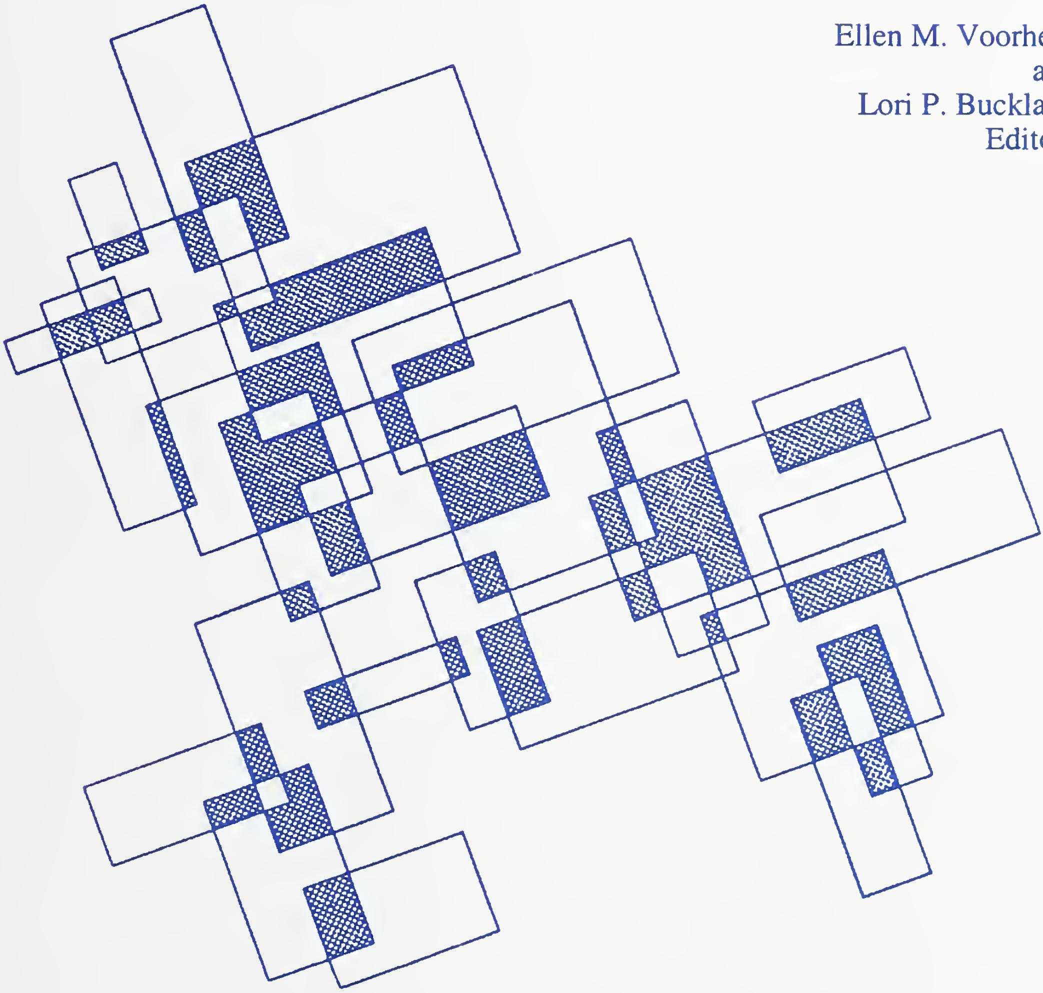
NIST Special Publication 500-251

Information Technology:

The Eleventh Text REtrieval Conference, TREC 2002



Ellen M. Voorhees
and
Lori P. Buckland
Editors

NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

The National Institute of Standards and Technology was established in 1988 by Congress to "assist industry in the development of technology . . . needed to improve product quality, to modernize manufacturing processes, to ensure product reliability . . . and to facilitate rapid commercialization . . . of products based on new scientific discoveries."

NIST, originally founded as the National Bureau of Standards in 1901, works to strengthen U.S. industry's competitiveness; advance science and engineering; and improve public health, safety, and the environment. One of the agency's basic functions is to develop, maintain, and retain custody of the national standards of measurement, and provide the means and methods for comparing standards used in science, engineering, manufacturing, commerce, industry, and education with the standards adopted or recognized by the Federal Government.

As an agency of the U.S. Commerce Department's Technology Administration, NIST conducts basic and applied research in the physical sciences and engineering, and develops measurement techniques, test methods, standards, and related services. The Institute does generic and precompetitive work on new and advanced technologies. NIST's research facilities are located at Gaithersburg, MD 20899, and at Boulder, CO 80303. Major technical operating units and their principal activities are listed below. For more information visit the NIST Web site at http://www.nist.gov, or contact the Publications and Program Inquiries Desk, 301-975-3058.

Office of the Director

- National Quality Program
- International and Academic Affairs

Technology Services

- Standards Services
- Technology Partnerships
- Measurement Service
- Information Services
- Weights and Measures

Advanced Technology Program

- Economic Assessment
- Information Technology and Applications
- Chemistry and Life Sciences
- Electronics and Photonics Technology

Manufacturing Extension Partnership Program

- Regional Programs
- National Programs
- Program Development

Electronics and Electrical Engineering Laboratory

- Microelectronics
- Law Enforcement Standards
- Electricity
- Semiconductor Electronics
- Radio-Frequency Technology [1]
- Electromagnetic Technology[1]
- Optoelectronics[1]
- Magnetic Technology[1]

Manufacturing Engineering Laboratory

- Precision Engineering
- Manufacturing Metrology
- Intelligent Systems
- Fabrication Technology
- Manufacturing Systems Integration

Chemical Science and Technology Laboratory

- Biotechnology
- Process Measurements
- Surface and Microanalysis Science
- Physical and Chemical Properties[2]
- Analytical Chemistry

Physics Laboratory

- Electron and Optical Physics
- Atomic Physics
- Optical Technology
- Ionizing Radiation
- Time and Frequency[1]
- Quantum Physics[1]

Materials Science and Engineering Laboratory

- Intelligent Processing of Materials
- Ceramics
- Materials Reliability[1]
- Polymers
- Metallurgy
- NIST Center for Neutron Research

Building and Fire Research Laboratory

- Applied Economics
- Materials and Construction Research
- Building Environment
- Fire Research

Information Technology Laboratory

- Mathematical and Computational Sciences[2]
- Advanced Network Technologies
- Computer Security
- Information Access
- Convergent Information Systems
- Information Services and Computing
- Software Diagnostics and Conformance Testing
- Statistical Engineering

[1] At Boulder, CO 80303.

[2] Some elements at Boulder, CO.

NIST Special Publication 500-251

Information Technology: The Eleventh Text REtrieval Conference, TREC 2002

Ellen M. Voorhees and
Lori P. Buckland
Editors

Information Technology Laboratory
Information Access Division
National Institute of Standards and Technology
Gaithersburg, MD 20899-8940

May 2003

U.S. Department of Commerce
Donald L. Evans, Secretary

Technology Administration
Phillip J. Bond, Under Secretary of Commerce for Technology

National Institute of Standards and Technology
Arden L. Bement, Jr., Director

Reports on Information Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) stimulates U.S. economic growth and industrial competitiveness through technical leadership and collaborative research in critical infrastructure technology, including tests, test methods, reference data, and forward-looking standards, to advance the development and productive use of information technology. To overcome barriers to usability, scalability, interoperability, and security in information systems and networks, ITL programs focus on a broad range of networking, security, and advanced information technologies, as well as the mathematical, statistical, and computational sciences. This Special Publication 500-series reports on ITL's research in tests and test methods for information technology, and its collaborative activities with industry, government, and academic organizations.

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

National Institute of Standards and Technology
Special Publication 500-251
Natl. Inst. Stand. Technol.
Spec. Publ. 500-251
1175 pages (May 2003)
CODEN: NSPUE2

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 2003

For sale by the Superintendent of Documents, U.S. Government Printing Office
Internet: bookstore.gpo.gov — Phone: (202) 512-1800 — Fax: (202) 512-2250
Mail: Stop SSOP, Washington, DC 20402-0001

Foreword

This report constitutes the proceedings of the 2002 edition of the Text REtrieval Conference, TREC 2002, held in Gaithersburg, Maryland, November 19–22, 2002. The conference was co-sponsored by the National Institute of Standards and Technology (NIST), the Advanced Research and Development Activity (ARDA), and the Defense Advanced Research Projects Agency (DARPA). Approximately 175 people attended the conference, including representatives from 21 different countries. The conference was the eleventh in an on-going series of workshops to evaluate new technologies for text retrieval and related information-seeking tasks. Ninety-three groups submitted retrieval results to one or more of the workshop's tracks.

The workshop included plenary sessions, discussion groups, a poster session, and demonstrations. Because the participants in the workshop drew on their personal experiences, they sometimes cite specific vendors and commercial products. The inclusion or omission of a particular company or product implies neither endorsement nor criticism by NIST. Any opinions, findings, and conclusions or recommendations expressed in the individual papers are the authors' own and do not necessarily reflect those of the sponsors.

The sponsorship of the U.S. Department of Defense is gratefully acknowledged, as is the tremendous work of the program committee and the track coordinators.

Ellen Voorhees
April 7, 2003

TREC 2002 Program Committee

Ellen Voorhees, NIST, chair
James Allan, University of Massachusetts at Amherst
Nick Belkin, Rutgers University
Chris Buckley, Sabir Research, Inc.
Jamie Callan, Carnegie Mellon University
Gordon Cormack, University of Waterloo
Susan Dumais, Microsoft
Fred Gey, University of California at Berkeley
Donna Harman, NIST
David Hawking, CSIRO
Bill Hersh, Oregon Health & Science University
James Mayfield, APL, Johns Hopkins University
John Prange, U.S. Department of Defense
Steve Robertson, Microsoft
Karen Sparck Jones, University of Cambridge, UK
Ross Wilkinson, CSIRO

TABLE OF CONTENTS

PAPERS

APPENDIX

INDEX OF TREC 2002 PAPERS BY TASK/TRACK

Cross-Language

Filtering

Interactive

Novelty

Question Answering

Video

Carnegie Mellon University

CLIPS-IMAG

Columbia University

CWI

Dublin City University

Institut Eurécom

Fudan University

IBM Alamaden Research Center

IBM India Research Lab

IBM T.J. Watson Research Center

Indiana University

Imperial College London

Mediamill/ISIS University of Amsterdam

Mediamill TNO-TPD

Media Team Oulu

Web

Fondazione Ugo Bordoni

Fudan University

Hummingbird

IBM Research Labs Haifa

Illinois Institute of Technology

IRIT-SIG

Kasetsart University

Laboratories for Information Technology (LIT)

Queens College, CUNY

RMIT University

Technion

Tsinghua University

University of Amsterdam

University of Glasgow

University of Illinois at Chicago

University of Illinois at Urbana-Champaign

Abstract

This report constitutes the proceedings of the 2002 edition of the Text REtrieval Conference, TREC 2002, held in Gaithersburg, Maryland, November 19–22, 2002. The conference was co-sponsored by the National Institute of Standards and Technology (NIST), the Advanced Research and Development Activity (ARDA), and the Defense Advanced Research Projects Agency (DARPA). Ninety-three groups including participants from 21 different countries were represented.

TREC 2002 is the latest in a series of workshops designed to foster research in text retrieval and related technologies. This year's conference consisted of seven different tasks: cross-language retrieval, filtering, interactive retrieval, novelty detection, question answering, content-based access to video, and web-based retrieval.

The conference included paper sessions and discussion groups. This proceedings includes papers from most of the participants (some groups did not submit papers), track reports that define the problem addressed by the track plus summarize the main track results, and tables of individual group results. The TREC 2002 proceedings web site also contains system descriptions that detail the timing and storage requirements of the different runs.

Overview of TREC 2002

Ellen M. Voorhees
National Institute of Standards and Technology
Gaithersburg, MD 20899

1 Introduction

The eleventh Text REtrieval Conference, TREC 2002, was held at the National Institute of Standards and Technology (NIST) November 19–22, 2002. The conference was co-sponsored by NIST, the Information Awareness Office of the Defense Advanced Research Projects Agency (DARPA/IAO), and the US Department of Defense Advanced Research and Development Activity (ARDA).

TREC 2002 is the latest in a series of workshops designed to foster research on technologies for information retrieval. The workshop series has four goals:

- to encourage retrieval research based on large test collections;
- to increase communication among industry, academia, and government by creating an open forum for the exchange of research ideas;
- to speed the transfer of technology from research labs into commercial products by demonstrating substantial improvements in retrieval methodologies on real-world problems; and
- to increase the availability of appropriate evaluation techniques for use by industry and academia, including development of new evaluation techniques more applicable to current systems.

TREC 2002 contained seven areas of focus called "tracks". These included the Cross-Language Retrieval Track, the Filtering Track, the Interactive Retrieval Track, the Novelty Track, the Question Answering Track, the Video Retrieval Track, and the Web Retrieval Track. This was the first year for the novelty track, which fostered research into detecting redundant information within a relevant document set. The other tracks were run in previous TRECs, though the particular tasks performed in some of the tracks changed for TREC 2002.

Table 1 lists the 93 groups that participated in TREC 2002. The participating groups come from 21 different countries and include academic, commercial, and government institutions.

This paper serves as an introduction to the research described in detail in the remainder of the volume. The next section provides a summary of the retrieval background knowledge that is assumed in the other papers. Section 3 presents a short description of each track—a more complete description of a track can be found in that track's overview paper in the proceedings. The final section looks forward to future TREC conferences.

2 Information Retrieval

Information retrieval is concerned with locating information that will satisfy a user's information need. Traditionally, the emphasis has been on text retrieval: providing access to natural language texts where the set of documents to be searched is large and topically diverse. There is increasing interest, however, in finding appropriate information regardless of the medium that happens to contain that information. Thus "document" can be interpreted as any unit of information such as a web page or a video clip.

The prototypical retrieval task is a researcher doing a literature search in a library. In this environment the retrieval system knows the set of documents to be searched (the library's holdings), but cannot anticipate the particular topic that will be investigated. We call this an *ad hoc* retrieval task, reflecting the arbitrary

Table 1: Organizations participating in TREC 2002

Ajou University
Alicante University
BBN Technologies
Carnegie Mellon U. (3 groups)
Chinese Academy of Sciences
City University, London
Clairvoyance Corp.
CLIPS-IMAG
CL Research
Columbia University (2 groups)
CSIRO
CWI, The Netherlands
Dublin City University
Fudan University
Hummingbird
IBM-Haifa
IBM-T.J. Watson (3 groups)
Illinois Institute of Technology
Imperial College of Science, Tech. & Medicine
Indiana University
InsightSoft-M
Institut EURECOM
IRIT/SIG
ITC-irst
Johns Hopkins University, APL
Kasetsart University
KerMIT Consortium
Laboratory for Information Technology, Singapore
Language Computer Corp.
David Lewis
LIMSI
Massachusetts Institute of Technology
Microsoft Research Asia
Microsoft Research Ltd.
The MITRE Corp.
Moscow Medical Academy
National Institute of Informatics
National Taiwan University
National University of Singapore (2 groups)
NTT Communication Science Labs
Oregon Health and Science University
Pohang University of Science and Technology
Prous Science
Queens College, CUNY
Queensland University of Technology
RMIT
Rutgers University (2 groups)
StreamSage, Inc.
Syracuse University
Tampere University of Technology
TNO TPD, The Netherlands
Tokyo University of Science
Tsinghua University
Université d'Angers
Université de Montreal
University of Amsterdam (2 groups)
University of Avignon
University of Bremen
University of Buffalo
University of California, Berkeley
University of Glasgow
University of Hertfordshire
University of Illinois at Chicago
University of Illinois at Urbana/Champaign
University of Iowa
University of Limerick
University of Maryland, Baltimore County
University of Maryland, College Park (2 groups)
University of Massachusetts
University of Melbourne
University of Michigan
University of Neuchatel
University of North Carolina, Chapel Hill
University of North Texas
University of Oulu
University of Pisa
University of Sheffield
University of Southern California, ISI
University of Sunderland
University of Toronto
University of Twente
University of Waterloo
University of York
Yonsei University and ETRI

subject of the search and its short duration. Other examples of ad hoc searches are web surfers using Internet search engines, lawyers performing patent searches or looking for precedences in case law, and analysts searching archived news reports for particular events. A retrieval system's response to an ad hoc search is generally a list of documents ranked by decreasing similarity to the query.

A *known-item search* is similar to an ad hoc search but the target of the search is a particular document (or a small set of documents) that the searcher knows to exist in the collection and wants to find again. Once again, the retrieval system's response is usually a ranked list of documents, and the system is evaluated by the rank at which the target document is retrieved.

In a document routing or *filtering* task, the topic of interest is known and stable, but the document collection is constantly changing [1]. For example, an analyst who wishes to monitor a news feed for items on a particular subject requires a solution to a filtering task. The filtering task generally requires a retrieval system to make a binary decision whether to retrieve each document in the document stream as the system sees it. The retrieval system's response in the filtering task is therefore an unordered set of documents (accumulated over time) rather than a ranked list.

Information retrieval has traditionally focused on returning entire documents that contain answers to questions rather than returning the answers themselves. This emphasis is both a reflection of retrieval systems' heritage as library reference systems and an acknowledgement of the difficulty of question answering. However, for certain types of questions, users would much prefer the system to answer the question than be forced to wade through a list of documents looking for the specific answer. To encourage research on systems that return answers instead of document lists, TREC has had a question answering track since 1999.

2.1 Test collections

Text retrieval has a long history of using retrieval experiments on test collections to advance the state of the art [3, 6, 9], and TREC continues this tradition. A test collection is an abstraction of an operational retrieval environment that provides a means for researchers to explore the relative benefits of different retrieval strategies in a laboratory setting. Test collections consist of three parts: a set of documents, a set of information needs (called *topics* in TREC), and *relevance judgments*, an indication of which documents should be retrieved in response to which topics.

2.1.1 Documents

The document set of a test collection should be a sample of the kinds of texts that will be encountered in the operational setting of interest. It is important that the document set reflect the diversity of subject matter, word choice, literary styles, document formats, etc. of the operational setting for the retrieval results to be representative of the performance in the real task. Frequently, this means the document set must be large. The primary TREC test collections contain about 2 gigabytes of text (between 500,000 and 1,000,000 documents). The document sets used in various tracks have been smaller and larger depending on the needs of the track and the availability of data.

The primary TREC document sets consist mostly of newspaper or newswire articles, though there are also some government documents (the *Federal Register*, patent applications) and computer science abstracts (*Computer Selects* by Ziff-Davis publishing) included. High-level structures within each document are tagged using SGML, and each document is assigned an unique identifier called the DOCNO. In keeping of the spirit of realism, the text was kept as close to the original as possible. No attempt was made to correct spelling errors, sentence fragments, strange formatting around tables, or similar faults.

2.1.2 Topics

TREC distinguishes between a statement of information need (the topic) and the data structure that is actually given to a retrieval system (the query). The TREC test collections provide topics to allow a wide range of query construction methods to be tested and also to include a clear statement of what criteria make a document relevant. The format of a topic statement has evolved since the beginning of TREC, but it has been stable for the past several years. A topic statement generally consists of four sections: an identifier, a title, a description, and a narrative. An example topic taken from this year's filtering track is shown in figure 1.

The different parts of the TREC topics allow researchers to investigate the effect of different query lengths on retrieval performance. The "titles" in topics 301–450 were specially designed to allow experiments with very short queries; those title fields consist of up to three words that best describe the topic. The description field is a one sentence description of the topic area. The narrative gives a concise description of what makes a document relevant.

```
<num> Number:  R111
<title> Telemarketing practices U.S.

<desc> Description:
Find documents which reflect telemarketing practices in the U.S. which are intrusive or
deceptive and any efforts to control or regulate against them.
<narr> Narrative:
Telemarketing practices found to be abusive, intrusive, evasive, deceptive, fraudulent,
or in any way unwanted by persons contacted are relevant.  Only such practices in the U.S.
are relevant.  All efforts to halt these practices, including lawsuits, legislation or
regulation are also relevant.
```

Figure 1: A sample TREC 2002 topic from the filtering track.

Participants are free to use any method they wish to create queries from the topic statements. TREC distinguishes among two major categories of query construction techniques, automatic methods and manual methods. An automatic method is a means of deriving a query from the topic statement with no manual intervention whatsoever; a manual method is anything else. The definition of manual query construction methods is very broad, ranging from simple tweaks to an automatically derived query, through manual construction of an initial query, to multiple query reformulations based on the document sets retrieved. Since these methods require radically different amounts of (human) effort, care must be taken when comparing manual results to ensure that the runs are truly comparable.

TREC topic statements are created by the same person who performs the relevance assessments for that topic (the *assessor*). Usually, each assessor comes to NIST with ideas for topics based on his or her own interests, and searches the document collection using NIST's PRISE system to estimate the likely number of relevant documents per candidate topic. The NIST TREC team selects the final set of topics from among these candidate topics based on the estimated number of relevant documents and balancing the load across assessors.

2.1.3 Relevance judgments

The relevance judgments are what turns a set of documents and topics into a test collection. Given a set of relevance judgments, the retrieval task is then to retrieve all of the relevant documents and none of the irrelevant documents. TREC almost always uses binary relevance judgments—either a document is relevant to the topic or it is not. To define relevance for the assessors, the assessors are told to assume that they are writing a report on the subject of the topic statement. If they would use any information contained in the document in the report, then the (entire) document should be marked relevant, otherwise it should be marked irrelevant. The assessors are instructed to judge a document as relevant regardless of the number of other documents that contain the same information.

Relevance is inherently subjective. Relevance judgments are known to differ across judges and for the same judge at different times [7]. Furthermore, a set of static, binary relevance judgments makes no provision for the fact that a real user's perception of relevance changes as he or she interacts with the retrieved documents. Despite the idiosyncratic nature of relevance, test collections are useful abstractions because the *comparative* effectiveness of different retrieval methods is stable in the face of changes to the relevance judgments [10].

The relevance judgments in early retrieval test collections were complete. That is, a relevance decision was made for every document in the collection for every topic. The size of the TREC document sets makes complete judgments utterly infeasible—with 800,000 documents, it would take over 6500 hours to judge the entire document set for one topic, assuming each document could be judged in just 30 seconds. Instead, TREC uses a technique called pooling [8] to create a subset of the documents (the "pool") to judge for a topic. Each document in the pool for a topic is judged for relevance by the topic author. Documents that are not in the pool are assumed to be irrelevant to that topic.

The judgment pools are created as follows. When participants submit their retrieval runs to NIST, they rank their runs in the order they prefer them to be judged. NIST chooses a number of runs to be merged into the pools, and selects that many runs from each participant respecting the preferred ordering. For each selected run, the top X documents (usually, $X = 100$) per topic are added to the topics' pools. Since the retrieval results are ranked by decreasing similarity to the query, the top documents are the documents most likely to be relevant to the topic. Many documents are retrieved in the top X for more than one run, so the pools are generally much smaller the theoretical maximum of X × *the-number-of-selected-runs* documents (usually about 1/3 the maximum size).

The use of pooling to produce a test collection has been questioned because unjudged documents are assumed to be not relevant. Critics argue that evaluation scores for methods that did not contribute to the pools will be deflated relative to methods that did contribute because the non-contributors will have highly ranked unjudged documents.

Zobel demonstrated that the quality of the pools (the number and diversity of runs contributing to the pools and the depth to which those runs are judged) does affect the quality of the final collection [12]. He also found that the TREC collections were not biased against unjudged runs. In this test, he evaluated each run that contributed to the pools using both the official set of relevant documents published for that collection and the set of relevant documents produced by removing the relevant documents uniquely retrieved by the run being evaluated. For the TREC-5 ad hoc collection, he found that using the unique relevant documents increased a run's 11 point average precision score by an average of 0.5 %. The maximum increase for any run was 3.5 %. The average increase for the TREC-3 ad hoc collection was somewhat higher at 2.2 %.

A similar investigation of the TREC-8 ad hoc collection showed that every automatic run that had a mean average precision score of at least .1 had a percentage difference of less than 1 % between the scores with and without that group's uniquely retrieved relevant documents [11]. That investigation also showed that the quality of the pools is significantly enhanced by the presence of recall-oriented manual runs, an effect noted by the organizers of the NTCIR (NACSIS Test Collection for evaluation of Information Retrieval systems) workshop who performed their own manual runs to supplement their pools [5].

While the lack of any appreciable difference in the scores of submitted runs is not a guarantee that all relevant documents have been found, it is very strong evidence that the test collection is reliable for comparative evaluations of retrieval runs. Indeed, the differences in scores resulting from incomplete pools observed here are smaller than the differences that result from using different relevance assessors [10].

2.2 Evaluation

Retrieval runs on a test collection can be evaluated in a number of ways. In TREC, all ad hoc tasks are evaluated using the `trec_eval` package written by Chris Buckley of Sabir Research [2]. This package reports about 85 different numbers for a run, including *recall* and *precision* at various cut-off levels plus single-valued summary measures that are derived from recall and precision. Precision is the proportion of retrieved documents that are relevant, while recall is the proportion of relevant documents that are retrieved. A cut-off level is a rank that defines the retrieved set; for example, a cut-off level of ten defines the retrieved set as the top ten documents in the ranked list. The `trec_eval` program reports the scores as averages over the set of topics where each topic is equally weighted. (The alternative is to weight each relevant document equally and thus give more weight to topics with more relevant documents. Evaluation of retrieval effectiveness historically weights topics equally since all users are assumed to be equally important.)

Precision reaches its maximal value of 1.0 when only relevant documents are retrieved, and recall reaches its maximal value (also 1.0) when all the relevant documents are retrieved. Note, however, that these theoretical maximum values are not obtainable as an average over a set of topics at a single cut-off level because different topics have different numbers of relevant documents. For example, a topic that has fewer than ten relevant documents will have a precision score less than one after ten documents are retrieved regardless of how the documents are ranked. Similarly, a topic with more than ten relevant documents must have a recall score less than one after ten documents are retrieved. At a single cut-off level, recall and precision reflect the same information, namely the number of relevant documents retrieved. At varying cut-off levels, recall and precision tend to be inversely related since retrieving more documents will usually increase recall while degrading precision and vice versa.

Of all the numbers reported by `trec_eval`, the recall-precision curve and mean (non-interpolated) average precision are the most commonly used measures to describe TREC retrieval results. A recall-precision curve plots precision as a function of recall. Since the actual recall values obtained for a topic depend on the number of relevant documents, the average recall-precision curve for a set of topics must be interpolated to a set of standard recall values. The particular interpolation method used is given in Appendix A, which also defines many of the other evaluation measures reported by `trec_eval`. Recall-precision graphs show the behavior of a retrieval run over the entire recall spectrum.

Mean average precision is the single-valued summary measure used when an entire graph is too cumbersome. The average precision for a single topic is the mean of the precision obtained after each relevant document is retrieved (using zero as the precision for relevant documents that are not retrieved). The mean average precision for a run consisting of multiple topics is the mean of the average precision scores of each of the individual topics in the run. The average precision measure has a recall component in that it reflects the performance of a retrieval run across all relevant documents, and a precision component in that it weights documents retrieved earlier more heavily than documents retrieved later. Geometrically, mean average precision is the area underneath a non-interpolated recall-precision curve.

Only three of the tasks in TREC 2002, the topic distillation task in the web track, the routing task in the filtering track, and the task in the cross-language track, were tasks that can be evaluated with `trec_eval`. The remaining tasks used other evaluation measures that are described in detail in the track overview paper for that task, and are briefly described in Appendix A. The bulk of Appendix A consists of the evaluation output for each run submitted to TREC 2002.

3 TREC 2002 Tracks

TREC's track structure was begun in TREC-3 (1994). The tracks serve several purposes. First, tracks act as incubators for new research areas: the first running of a track often defines what the problem *really* is, and a track creates the necessary infrastructure (test collections, evaluation methodology, etc.) to support research on its task. The tracks also demonstrate the robustness of core retrieval technology in that the same techniques are frequently appropriate for a variety of tasks. Finally, the tracks make TREC attractive to a broader community by providing tasks that match the research interests of more groups.

Table 2 lists the different tracks that were in each TREC, the number of groups that submitted runs to that track, and the total number of groups that participated in each TREC. The tasks within the tracks offered for a given TREC have diverged as TREC has progressed. This has helped fuel the growth in the number of participants, but has also created a smaller common base of experience among participants since each participant tends to submit runs to fewer tracks.

This section describes the tasks performed in the TREC 2002 tracks. See the track reports elsewhere in this proceedings for a more complete description of each track.

3.1 The Cross-Language (CLIR) track

The task in the CLIR track is an ad hoc retrieval task in which the documents are in one language and the topics are in a different language. The goal of the track is to facilitate research on systems that are able to retrieve relevant documents regardless of the language a document happens to be written in. The TREC 2002 cross-language track used Arabic documents and English topics. An Arabic version of the topics was also developed so that cross-language retrieval performance could be compared with the equivalent monolingual performance.

The document set was created and released by the Linguistic Data Consortium ("Arabic Newswire Part 1", catalog number LDC2001T55); it is the same document collection that was used in the TREC 2001 CLIR track. The collection consists of 869 megabytes of news articles taken from the Agence France Presse (AFP) Arabic newswire: 383,872 articles dated from May 13, 1994 through December 20, 2000.

Fifty topics were created for the track using the standard topic development protocol except that topic development took place at the Linguistic Data Consortium (LDC). The assessors were fluent in both Arabic and English (for most assessors Arabic was their first language). They searched the document collection using a retrieval system developed by the LDC for the task and Arabic as the query language. Once fifty topics were selected from among the candidate topics, the assessor who developed the topic created the full topic statement first in English and then in Arabic. The assessors' instructions were that the Arabic

Table 2: Number of participants per track and total number of distinct participants in each TREC

Track	TREC										
	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002
Ad Hoc	18	24	26	23	28	31	42	41	—	—	—
Routing	16	25	25	15	16	21	—	—	—	—	—
Interactive	—	—	3	11	2	9	8	7	6	6	6
Spanish	—	—	4	10	7	—	—	—	—	—	—
Confusion	—	—	—	4	5	—	—	—	—	—	—
Database Merging	—	—	—	3	3	—	—	—	—	—	—
Filtering	—	—	—	4	7	10	12	14	15	19	21
Chinese	—	—	—	—	9	12	—	—	—	—	—
NLP	—	—	—	—	4	2	—	—	—	—	—
Speech	—	—	—	—	—	13	10	10	3	—	—
Cross-Language	—	—	—	—	—	13	9	13	16	10	9
High Precision	—	—	—	—	—	5	4	—	—	—	—
Very Large Corpus	—	—	—	—	—	—	7	6	—	—	—
Query	—	—	—	—	—	—	2	5	6	—	—
Question Answering	—	—	—	—	—	—	—	20	28	36	34
Web	—	—	—	—	—	—	—	17	23	30	23
Video	—	—	—	—	—	—	—	—	—	12	19
Novelty	—	—	—	—	—	—	—	—	—	—	13
Total participants	22	31	33	36	38	51	56	66	69	87	93

version of the topic should contain the same information as the English version, but should be expressed in a way that would seem natural to a native speaker of Arabic. The English and Arabic versions of the topics were made available to the track participants who were asked to check the topics for substantive differences among the different versions. A few changes were suggested by participants, and those changes were made to produce the final version of the topics.

Forty-one runs from nine different groups were submitted to the track. Twenty-three of the runs were cross-language runs and eighteen were monolingual runs. One monolingual run was a manual run. Two groups submitted only monolingual runs, one group submitted only cross-language runs, and the remaining six groups submitted at least one run of each type.

The assessment pools were created using all submitted runs and using the top 100 documents from each run. The average size of the pools was 769 documents. The LDC assessors judged each document in the pools using binary (relevant/not relevant) assessments.

There was some concern over the test collection built in the TREC 2001 CLIR track in that the judgment pools were not as complete as they ideally would be. The 2001 collection contained 25 topics. For 13 of the topics, at least 40 % of the known relevant documents for that topic were retrieved by one group. Further, mean average precision scores decreased by an average of 8 %, with a maximum difference of 21 %, when runs were evaluated without using the group's unique relevant documents. This year's collection has no similar concerns. The average number of relevant documents over the 50 topics is 118.2, with a minimum of 3 relevant documents and a maximum of 523 relevant documents. Only 5 topics had at least 40 % of the relevant documents retrieved by one group. Changes in mean average precision scores when unique relevant documents are removed were similar to the TREC ad hoc collections: an average decrease of a little less than 2 % with a maximum change of 5.7 %.

The average size of the TREC 2001 pools was larger then the average size of the 2002 pools (164.9 vs. 118.2) even though the 2002 pools used more runs and went deeper into the ranked list. Thus, the results produced by different systems are clearly more similar to one another in 2002 than in 2001. But why this should be so is unclear. It could be that the systems are converging to a single effective strategy. The track made a standard set of resources such as stemmers and bi-lingual dictionaries available to participants; common resources are likely to reduce differences among systems. It may also be that the topic set in 2002 was intrinsically easier than the 2001 set, though the effectiveness of the best automatic systems is slightly lower in 2002 than in 2001 which would suggest the opposite conclusion.

As has become common in the CLIR track, the most effective runs as measured by mean average precision (MAP) were cross-language runs, not monolingual runs. The best cross-language run was from the University of Massachusetts, run UMassX6n, with a MAP score of .3996, while the best monolingual run was from the University of Neuchatel, run UniNE3, with a MAP score of .3807. These two runs were the top two runs as measured by precision at document cut-off level 10 as well, but in this case their order was reversed: `UMassX6n` had a P(10) score of.488 and `UniNE3` a score of .516. The University of Massachusetts submitted one monolingual run (MAP: .3619, P(10): .432) though there is not a corresponding cross-language run from the University of Neuchatel. Thus it is not possible to tell from this data whether monolingual access is better for high precision searches in general.

3.2 The Filtering track

The filtering task is to retrieve just those documents in a document stream that match the user's interest as represented by the topic. Once again there were three tasks in the TREC 2002 filtering track: an adaptive filtering task, a batch filtering task, and a routing task.

In the adaptive filtering task, a system starts with a profile derived from the topic statement and a small number of examples of relevant documents, and processes documents one at a time in date order. For each document in turn, the system must make a binary decision whether to retrieve it. If the system decides to retrieve the document, it obtains the relevance judgment for that document, and can modify its profile based on the judgment if desired. The final output is the unranked set of retrieved documents for the topic.

The batch filtering task is a simpler version of the adaptive task. In this task, the system is given a topic and a (relatively large) set of training documents such that each document in the training set is labeled as relevant or not relevant. From this data, the system creates a profile and a rule for when a document should be retrieved. The rule is applied to each document in the test set of documents without further modification. Once again, the final output is an unranked set of retrieved documents.

In the *routing* task, the system again builds a profile or query from a topic statement and a training set of documents, but then uses the query to rank the test portion of the collection. Ranking the collection by similarity to the query (routing) is an easier problem than making a binary decision as to whether a document should be retrieved (batch filtering) because the latter requires a threshold that is difficult to set appropriately. The final output for the routing task is a list of 1000 documents ranked by decreasing similarity to the query.

The TREC 2002 filtering task used the same corpus as the TREC 2001 track, "Reuters Corpus, Volume 1, English language, 1996-08-20 to 1997-08-19" from Reuters (`http://about.reuters.com/researchandstandards/corpus/`). This collection consists of approximately 810,000 news stories from August 20, 1996 through August 19, 1997. Each document is tagged with Reuters category codes, and a hierarchy of the Reuters category codes is included with the corpus.

Two distinct types of topics were created for the track. The first set of 50 topics was created by NIST assessors using the standard topic development protocol. Once a candidate topic was provisionally accepted, the author of the topic was given five document sets of approximately 100 documents each to judge for the topic. These document sets were created at NIST by using the relevant documents found in earlier rounds as input to a small set of different feedback systems. The combined set of judged documents was used as the training data for that topic. A second set of 50 topics was created by defining a topic to be the intersection of pairs of Reuters category codes. In this case, a document is relevant to the topic if it has been assigned both of the appropriate category labels. The topic statement for an intersection topic is a simple combination of the category descriptors. A filtering track run was required to process all 100 topics.

Since filtering runs do not produce a ranked list, they cannot be evaluated using the usual IR measures. Instead, filtering runs are generally evaluated using a utility function whereby a system is rewarded some number of points for retrieving a relevant document and penalized a different number of points for retrieving an irrelevant document. Because raw utility scores do not average well, the scores for individual topics are normalized, scaled, and then averaged. Details of the TREC 2002 filtering evaluation measures are given in the filtering track overview paper. Routing runs are evaluated using mean average precision since routing runs produce a ranked list of documents.

Seventy-three runs from twenty-one different groups were submitted to the filtering track. Of these, 40 runs are adaptive filtering runs, 16 are batch filtering runs, and 17 are routing runs. The most striking features of the filtering results was the large difference in effectiveness between the assessor-created topics and the intersection topics. System effectiveness was uniformly poor for the intersection topics, including those systems that did relatively well on the human constructed topics, and even including routing runs despite earlier research that shows the initial topic statement is a minor factor in system effectiveness for routing. The intersection topics were included in this year's test set to test whether this very inexpensive topic construction method is adequate for building comparative test collections. Until the reasons for the very large difference in effectiveness are understood, we must conclude that intersection topics are not good substitutes for information need statements.

3.3 The Interactive track

The interactive track was one of the first tracks to be introduced into TREC. Since its inception, the high-level goal of the track has been the investigation of searching as an interactive task by examining the process as well as the outcome.

The TREC 2002 track was the second year of a two-year plan to implement a metrics-based comparison of interactive systems as suggested by the SIGIR 2000 Workshop on Interactive Retrieval at TREC and Beyond [4]. In the first year of the plan during TREC 2001, participants performed observational studies of subjects using publicly-accessible tools and the live web to accomplish a search task. The TREC 2002 track followed the observational studies by laboratory experiments focusing on question answering using web data.

The track used an "open" version of the .GOV collection that was created for the TREC 2002 web track. The collection was open in the sense that some links to pages outside the collection could be followed. Most of the participants accessed the collection through the Panoptic search engine made available by CSIRO (see `http://www.panopticsearch.com/`).

The track defined eight different search tasks, two instances each for four general searching activities: looking for personal health information; seeing guidance on US government laws and policies; making travel plans; and gathering material for a report on a given subject. The search tasks were formulated such that the searcher was asked to find either any N short answers to the question or any N web sites that met the need specified in the task. The experimental protocol used in the track was based on the protocol developed for the TREC-9 interactive track and allows the comparison of two systems (or system variants). The protocol required a minimum of 16 searchers. Each searcher performed all eight tasks, half the task on one system and the other half on the other system. Searchers were given at least ten minutes to complete the search, and groups were required to report the results obtained after ten minutes.

Six groups participated in the interactive track. Each group examined their own set of hypotheses as suggested by their TREC 2001 observational studies. See the site reports in these proceedings for details of the individual experiments.

3.4 The Novelty track

The novelty track is a new track in TREC 2002. The goal of the track is to investigate systems' abilities to locate relevant and new (nonredundant) information within the ranked set of documents returned by a traditional document retrieval system. Similar to the question answering track, the motivation for the track is to assist the user of a retrieval system by eliminating extraneous information from the system response.

The data for the track was taken from TRECs 6–8 ad hoc collections. NIST selected 50 topics from that set and selected up to 25 relevant documents for each topic (if there were more than 25 relevant documents, the top 25 according to the document ranking used were selected; in all cases documents from the *Congressional Record* were eliminated). Each set of relevant documents was ranked at NIST using the ordering produced by an effective manual run from the appropriate TREC; participants were required to process the documents in this order. Each document was also split into sentences at NIST and sentences were assigned identifiers.

Table 3: Average F scores for baseline and system results for the Novelty track.

	Relevant	New
Second human judges	0.371	0.353
Random sentences	0.040	0.036
thunv1	0.235	0.217

A novelty track run consists of two ordered sets of sentence identifiers for each of the 50 topics. The first set of sentences is the set of sentences the system determined to contain relevant information. The second set of sentences (required to be a subset of the first set) is the set of sentences the system determined to contain new information, that is, relevant information that is not contained in earlier sentences.

Judgment data was created by having assessors manually perform the task. Each topic was independently judged by two different assessors so that the effects of different human opinions could be assessed. In general, the two different assessors did disagree, though much of the disagreement revolved around how much context to include in the relevant set. That is, one assessor would include a string of sequential sentences in the relevant set while the other assessor would select fewer sentences from the same general area of the document. Scoring for the track was based on the smaller relevant set (and its associated new set) because that seemed the best match for the task. Participants were told that the scoring would be based on the smaller set before runs were submitted, but, of course, they did not have access to the assessor sentence sets. One assessor disagreed with the original assessor's relevance judgments for topic 310 and could find no relevant sentences in any of the documents. We eliminated that topic from the final test set, so scores were computed over the remaining 49 topics.

The track guidelines specified sentence set recall and precision as the evaluation measures for the track. Let M be the number of matched sentences, i.e., the number of sentences selected by both the assessor and the system, A be the number of sentences selected by the assessor, and S be the number of sentences selected by the system. Then sentence set recall is M/A and precision is M/S. The F measure with recall and precision weighted equally (i.e., $\beta = 1$) was used as the final score for a topic.

Thirteen groups submitted 43 runs to the novelty track. For all runs, the F score for the relevant sentence sets was greater than the score for the new sentence sets. This might suggest that finding the relevant parts of a document is somewhat easier than finding the nonredundant parts, but is more likely to be a result of the different characteristics of the two tasks. A very small percentage of the total number of sentences were relevant (a median of 2 % across the 49 topics), whereas a very high percentage of the relevant sentences were novel (a median of 93 % across the 49 topics).

One of the requirements for a new track is to do sanity-checking of the evaluation itself. To this end, NIST computed the average F scores for the second human assessor sentence sets and for sets of sentences randomly selected from the target documents. The results are shown in Table 3, which also includes the scores for the most effective system run, run thunv3, for comparison. The scores for the best system falls in between the human and random performance, support for a claim that the evaluation is credible.

3.5 The Question Answering (QA) track

The question answering track addresses the problem of information overload by encouraging research into systems that return actual answers, as opposed to ranked lists of documents, in response to a question. The TREC 2002 track contained two different tasks, the main task and the list task. Both tasks were also run in TREC 2001, though there were significant differences in the task definitions between the two years.

Both tasks used a new document collection known as the AQUAINT Corpus of English News Text as the source of answers. This corpus is comprised of documents from three different sources: the AP newswire from 1998–2000, the New York Times newswire from 1998–2000, and the (English portion of the) Xinhua News Agency from 1996–2000. There are approximately 1,033,000 documents and 3 gigabytes of text in the collection. The corpus may be obtained from the Linguistic Data Consortium (www.ldc.upenn.edu) as catalog number LDC2002T31.

The main task was the focus of the track. As in previous years, participants received a set of fact-based, short-answer questions, and systems were to return an answer to each question along with the id of a document that supports that answer. In contrast to previous years, systems could return only one response per question, and text snippets containing the answer were not acceptable—systems were required to return nothing more or less than the answer itself. Questions were not guaranteed to have an answer in the collection. A system could return "NIL" as a response to indicate its belief that the collection did not contain an answer to the question.

The change to requiring exact answers was motivated by the belief that forcing systems to return precisely the answer is a necessary step in improving QA technology, not that it is a good idea for deployed QA systems. Whether an answer was exact was determined by the NIST assessor. Assessors judged each response by assigning it exactly one of the following judgments:

incorrect: the answer string returned by the system does not contain a correct answer or the answer is not responsive;

unsupported: the answer string contains a correct answer but the document returned does not support that answer;

non-exact: the answer string contains a correct answer and the document supports that answer, but the string contains more than just the answer or is missing bits of the answer;

correct: the answer string consists of exactly a correct answer and that answer is supported by the document returned.

Being "responsive" means such things as including units for quantitative responses (e.g., $20 instead of 20) and answering with regard to a famous entity itself rather than its replicas or imitations. Only the "correct" judgment was accepted for scoring purposes. NIL was counted as correct when no correct answer was known to exist in the collection for that question.

The test set of questions for the main task consists of 500 questions drawn from MSNSearch and AskJeeves logs. NIST fixed the spelling, punctuation, and sometimes the grammar of the questions selected to be in the final question set, but the content of the question was precisely what was in the log. (Some errors remained despite NIST's attempts to fix such mistakes; questions with errors remained in the test set.) Because it is impossible to know what kind of a response is desired for definition questions (e.g., *Who is Colin Powell? What are steroids?*) when there is no specific target user, none of this type of question was included in the test set. NIST made no other attempt to control the relative frequency of different question types. Forty-six of the questions have no known correct answer in the document collection.

Systems were required to return exactly one response per question. Within the submission file, the *questions* were ordered from most confident response to least confident response. That is, the question for which the system was most confident that it had returned a correct response was ranked first, then the question that the system was next most confident about, etc. so that the last question was the question for which the system was least confident in its response. The question ordering was done to test a system's ability to recognize whether it had found a good response since the final score assigned to a submission was based on this confidence ranking. The confidence-weighted score was inspired by the uninterpolated average precision measure for ranked retrieval output and is defined as

$$\frac{1}{500}\sum_{i=1}^{500}\frac{\text{number correct in first } i \text{ ranks}}{i}.$$

This measure rewards systems for answering questions correctly early in the ranking more than it rewards for answering questions correctly later in the ranking. (This is equivalent to penalizing systems more for incorrectly answering questions early in the ranking.)

Thirty-four different groups participated in the QA track. Each participant submitted at least one main task run for a total of 67 main task runs. The confidence-weighted evaluation measure succeeded in rewarding systems that were able to reliably determine whether they had found a good response, as illustrated in table 4. The table shows the number of questions whose answer was marked correct and the confidence-weighted score

Table 4: Number of questions answered correctly and confidence-weighted score for top 5 TREC 2002 main task QA runs.

Run	Number Correct	Confidence-weighted Score
LCCmain2002	415	0.856
exactanswer	271	0.691
pris2002	290	0.610
IRST02D1	192	0.589
IBMPQSQACYC	179	0.588

for the top five main task runs, ordered by confidence-score. The pris2002 run has a lower confidence score than the exactanswer run despite answering 19 additional questions correctly.

The list task required systems to assemble a set of answers as the response for a question. Each question asked for a given number of instances of a certain type. For example, one of the questions used in the track was *List 9 types of sweet potatoes.* The response to a list question was an unordered list of [*document-id, answer-string*] pairs, where each pair was treated as a single instance. As in the main task, answer-strings were required to be exact.

The questions for the list task were constructed by NIST assessors. The target number of instances to retrieve was selected such that the document collection contained more than the requested number of instances, but more than one document was required to meet the target. A single document could contain multiple instances, and the same instance might be repeated in multiple documents.

The assessors judged each list as a unit. Individual instances were judged as in the main task. In addition, the assessor also marked a set of instances as distinct. The assessor arbitrarily chose any one of a set of equivalent correct instances to mark as the distinct one, and marked the remainder as not distinct. The accuracy score for a list question was computed as the number of correct distinct instances retrieved divided by the number of requested instances. The score for a run was the average accuracy over the 25 questions in the test set.

Five groups submitted nine runs for the list task.

3.6 The Video track

TREC 2002 was the second year for the video track, a track designed to promote progress in content-based retrieval from digital video. This year's track contained three tasks: the shot boundary task, the feature extraction task, and the search task.

The video data for the track consisted of MPEG-1/VCD recordings from the Internet Archive (http://www.archive.org/movies) and the Open Video Project (http://www.open-video.org). The track defined a different set of files from these sources as the development sets and test sets for the different tasks. The search test collection contained approximately 40 hours of video, and the feature extraction and shot boundary test collections each contained about five hours of video. For the search and feature extraction tasks, the track also published a reference set of shot boundaries for the video collection. Runs for these two tasks returned lists of shots as defined by the reference set.

The goal in the shot boundary task was to (automatically) identify the shot boundaries in a given video clip. In addition to giving the location of the boundary as a time offset, systems were also required to specify whether the boundary was a cut or a gradual transition. System output was evaluated using automatic comparison to a set of reference shot boundaries created manually at NIST, using set recall and precision as the measures. Frame recall and frame precision (recall and precision of the individual frames within the shot) were also computed for each gradual transition detected by the system. Eight groups submitted 53 shot boundary runs.

There were two main motivations for the new feature extraction task. First, the ability to detect semantic

Outdoors: Segment contains a recognizably outdoor location.

Indoors: Segment contains a recognizably indoor location.

Face: Segment contains at least one human face with the nose, mouth, and both eyes visible.

People: Segment contains a group of two more humans, each of which is at least partially visible and is recognizable as a human.

Cityscape: Segment contains a recognizably city/urban/suburban setting.

Landscape: Segment contains a predominantly natural inland setting.

Text Overlay: Segment contains superimposed text large enough to be read.

Speech: A human voice uttering words is recognizable as such in this segment

Instrumental Sound: Sound produced by one or more musical instruments is recognizable as such in this segment.

Monologue: Segment contains an event in which a single person is at least partially visible and speaks for a long time without interruption by another speaker.

Figure 2: Descriptions of features to be detected in the Video track's feature extraction task.

concepts within video is seen as key to providing content-based access. The task is a first step toward building a benchmark for evaluating the effectiveness of particular feature detection methods. Second, the track implemented a plan whereby participants' extraction output for features specific to the search task were made available to other participants. This allowed the track to investigate methods for exploiting detected features in a search task.

Ten different features, shown in figure 2, were specified as test features. Shots containing the features were determined by NIST assessors (using pools of shots submitted by participants as for document relevance assessing). A shot contains a feature if at least one frame within the shot matches the feature's description, and otherwise does not contain the feature.

A feature extraction run consisted of a ranked list of up to 1000 shots ordered by likelihood that the shot contains the feature. Runs were evaluated using precision and recall, as well as uninterpolated average precision. Measures were computed for each feature individually, but not averaged across features. Eleven groups submitted 18 feature extraction runs.

The search task was a typical ad hoc retrieval task where the "documents" were video shots and the topics were multimedia statements of information need. A search task run consisted of a ranked list of the top 100 shots ordered by likelihood that the shot satisfies the topic. The track distinguished two type of runs: "manual" runs where a human formulated the query based on the topic but there was no further human intervention in the run, and "interactive" where a human formulated an initial query and then refined the query based on initial search output to form the final ranked list. Groups submitting interactive runs were required to report the amount of time the searcher spent to produce the final ranked list. Effectiveness was measured using traditional ranked retrieval measures.

The 25 test topics were created at NIST. The user model assumed during the topic creation process was that of a trained searcher trying to find material for reuse from a large video archive. Each topic statement included a brief textual description of the desired information (e.g., "Find shots containing Washington Square Park's arch in New York City.") and one or more examples of the desired information. Examples were references to video clips, still images, and audio clips. Twelve groups submitted 40 search task runs. Thirteen of the runs were interactive runs and 27 were manual runs.

3.7 The Web track

The goal in the web track is to investigate retrieval behavior when the collection to be searched is a large hyperlinked structure such as the World Wide Web. This year's track used a new document set and defined two new tasks. The topic distillation task is an ad hoc retrieval task in which the goal is to retrieve "key resources" rather than relevant documents. The named page finding task is a known-item task where the goal is to find a particular page that has been named by the user.

The document collection used for both tasks was a new collection known as the .GOV collection (http://www.ted.cmis.csiro.au/TRECWeb/govinfo.html) because it is based on a crawl of .gov web sites. The crawl occurred in January, 2002 and was made to mimic the way a real search service of the .gov pages might make a crawl. The crawl was breadth-first and stopped after one million html pages had been fetched. The crawl also included approximately 250,000 other types of pages (postscript, word, and pdf files) as well as images. The documents in the collection contain both page content and the information returned by the http daemon; text extracted from the non-html pages is also included in the collection. Charlie Clark of the University of Waterloo made the crawl using a machine made available by Ed Fox of Virginia Tech. The document collection was created from the crawl by CSIRO who are also distributing the collection.

The goal in the topic distillation task is to assemble a short, but comprehensive, list of pages that are good information resources on a particular topic. An example use for such a search is a user assembling a bookmark list for the target topic. Examples of key resources pages include the home page of a site dedicated to the topic; the main page of a sub-site dedicated to the topic; a highly useful single document (e.g., a postscript document) dedicated to the topic; a highly useful page of links (hub page) on the topic; and a relevant service such as a search engine dedicated to the topic.

NIST assessors created 50 topics for the topic distillation task. The topics are much like regular TREC topics, except the content targets topics for which the .GOV collection contains good key resources. Assessment was performed on pooled results as in a standard ad hoc task, but a document was judged as to whether it is a good resource pages, not whether the page is relevant. The main evaluation measure used for the task was precision at cut-off level 10 to focus the systems on retrieving a concise list of good resources.

The named page task is similar to the homepage finding task from TREC 2001 except the target page could be any page in the collection rather than an entry page to a site. The topics for the name page task were created by NIST assessors who searched the document collection looking for pages that were unique and that contained content a user might want to return to. For example, one topic asked for the directions to the Berkeley National Laboratory. Topics consisted of a single phrase, such as "directions Berkeley National Laboratory" for the example above.

The goal in the task is for a system to return the one target page for each topic. For evaluation, participants returned a ranked list of 50 documents per topic, and were scored using the mean reciprocal rank of the target page across the 150 test topics. Small pools consisting of the top 10 pages from each judged run were created to check for pages that had different DOCNOs but were equivalent pages (caused by mirroring and the like). The rank of the target page whose rank was closest to one was used as the score for each topic.

Twenty-three groups submitted a total 141 runs to the web track. Of those runs, 71 were topic distillation task runs and 70 were named page finding task runs. The results of the topic distillation task suggest there is still some question as to how exactly the task should be implemented for both assessors and participants. The web track in TREC 2003 will explore these questions in depth, including adding an interactive version of the task.

4 The Future

TREC 2003 will see significant changes in the tracks to be offered: several existing tracks will be suspended and new tracks introduced. The video track will be spun off into its own evaluation program to allow the effort to expand to include other facets of video retrieval. The new TRECVID[1] workshop will meet at NIST

[1] http://www-nlpir.nist.gov/projects/trecvid

immediately prior to TREC 2003. Since the NTCIR[2] and CLEF[3] evaluations provide venues for cross-language retrieval research, the Cross-Language track will be discontinued in TREC. The interactive track will not be a separate track in TREC 2003, but interactive subtasks will be incorporated into other tracks. In particular, the web track will have an interactive version of the topic distillation task in 2003. Finally, the filtering track will also not run in TREC 2003. Participants interested in the filtering task are encouraged to use the filtering track mailing list (see `http://trec.nist.gov/tracks.html`) to discuss plans for future tracks.

Three new tracks will be added to TREC 2003. The Genome track will provide a forum for the evaluation of information retrieval systems in the genomics domain. This first running of the track in 2003 follows an exploratory "pre-track" that occurred during 2002. Each of the remaining two new tracks will explore different aspects of ad hoc retrieval. The task in the Robust Retrieval track will be a traditional ad hoc task, but with an emphasis on individual topic effectiveness rather than average effectiveness. The goal of this track is to improve the consistency of retrieval technology by focusing on poorly performing topics. The goal of the HARD (Highly Accurate Retrieval from Documents) track is also to improve the effectiveness of ad hoc searches, but in this case the emphasis will be on customizing retrieval for individual users by exploiting information about the search context and using very targeted interaction with the searcher.

Acknowledgements

Special thanks to the track coordinators who make the variety of different tasks addressed in TREC possible.

References

[1] Nicholas J. Belkin and W. Bruce Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, December 1992.

[2] Chris Buckley. trec_eval IR evaluation package. Available from `ftp://ftp.cs.cornell.edu/pub/smart`.

[3] C. W. Cleverdon, J. Mills, and E. M. Keen. Factors determining the performance of indexing systems. Two volumes, Cranfield, England, 1968.

[4] William Hersh and Paul Over. SIGIR workshop on interactive retrieval at TREC and beyond. *SIGIR Forum*, 34(1):24–27, Spring 2000.

[5] Noriko Kando, Kazuko Kuriyama, Toshihiko Nozue, Koji Eguchi, Hiroyuki Kato, and Souichiro Hidaka. Overview of IR tasks at the first NTCIR workshop. In *Proceedings of the First NTCIR Workshop on Research in Japanese Text Retrieval and Term Recognition*, pages 11–44, 1999.

[6] G. Salton, editor. *The SMART Retrieval System: Experiments in Automatic Document Processing.* Prentice-Hall, Inc. Englewood Cliffs, New Jersey, 1971.

[7] Linda Schamber. Relevance and information behavior. *Annual Review of Information Science and Technology*, 29:3–48, 1994.

[8] K. Sparck Jones and C. van Rijsbergen. Report on the need for and provision of an "ideal" information retrieval test collection. British Library Research and Development Report 5266, Computer Laboratory, University of Cambridge, 1975.

[9] Karen Sparck Jones. *Information Retrieval Experiment.* Butterworths, London, 1981.

[10] Ellen M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information Processing and Management*, 36:697–716, 2000.

[2] `http://research.nii.ac.jp/ntcir`

[3] `http://clef.iei.pi.cnr.it`

[11] Ellen M. Voorhees and Donna Harman. Overview of the eighth Text REtrieval Conference (TREC-8). In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 1–24, 2000. NIST Special Publication 500-246. Electronic version available at `http://trec.nist.gov/pubs.html`.

[12] Justin Zobel. How reliable are the results of large-scale information retrieval experiments? In W. Bruce Croft, Alistair Moffat, C.J. van Rijsbergen, Ross Wilkinson, and Justin Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 307–314, Melbourne, Australia, August 1998. ACM Press, New York.

The TREC-2002 Arabic/English CLIR Track

Douglas W. Oard
College of Information Studies and Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742
oard@glue.umd.edu
and
Fredric C. Gey
UC Data Archive & Technical Assistance
University of California, Berkeley, CA 94720
gey@ucdata.berkeley.edu

Abstract

Nine teams participated in the TREC-2002 cross-language information retrieval track, which focused on retrieving Arabic language documents based on 50 topics that were originally prepared in English. Arabic translations of the topic descriptions were also made available to facilitate monolingual Arabic runs. This was the second year in which a large Arabic document collection was available. Three new teams joined the evaluation, and the cross-language aspect of the evaluation received more attention this year than in TREC-2001. A set of standard linguistic resources was made available to facilitate cross-system comparisons, and their use as a contrastive condition was encouraged. Unique contributions to the relevance pools were more typical of previous TREC evaluations then the results of TREC-2001 had been for the same document collection, with no run uniquely contributing more than 6% of the known relevant documents.

1 Introduction

The goal of the 2002 Text Retrieval Conference (TREC-2002) Cross-Language Information Retrieval (CLIR) task was to develop evaluation methodologies and evaluation resources to assess the effectiveness of ranked retrieval techniques that accept English queries and search Arabic documents. Monolingual Arabic experiments, in which both the queries and the documents were in Arabic, were also supported. Standard French translations were also provided in TREC-2001, but that was not done this year because no team expressed interest in working with French in preference to English. TREC-2002 was the ninth year in which non-English document retrieval has been evaluated at TREC, and the sixth year in which cross-language information retrieval has been the principal focus of that work. For a summary of prior evaluations, readers are referred to [1]. In this paper we describe the task and the evaluation collection, we summarize the techniques used by each participating team and briefly describe their results, and we offer some guidelines for future use of the TREC Arabic collection.

2 Task Description

As in past CLIR evaluations, the principal task for each group was to automatically build queries from topic descriptions written in one language (English, in this case) and then use those queries as a basis for ranking documents written in another language (Arabic, in this case) in order of decreasing degree (or probability) of topical relevance. Each participating team was allowed to submit as many as five runs for official scoring. In order to foster comparability, teams submitting cross-language runs were required to submit at least one run in which only the title and description fields of the topic description were used. Evaluation then proceeded by pooling the top 100 documents for each topic from each of the 41 submitted runs, manual examination each document in the pool by a human judge (usually the creator of the topic), and recording binary (yes/no) topical relevance judgments for each document in each topic's judgment pool. Participating teams were also invited to perform additional "post-hoc" runs, scoring them locally using relevance judgments provided by NIST, if they wished to investigate more conditions than would be possible using only the five official runs. The top 1000 documents in the ranked list for each topic was evaluated using a suite of measures. In this paper, we report the mean (over 50 topics) of the precision at 11 levels of recall and the mean (again, over 50 topics) of the uninterpolated average precision. Additional statistics for each run can be found elsewhere in this proceedings.

2.1 Topics

For TREC-2002, fifty topic descriptions (numbered AR26-AR75) were created in English in a collaborative process between the LDC and NIST (for TREC-2001, only 25 topics had been created). An example from this year's topic set is:

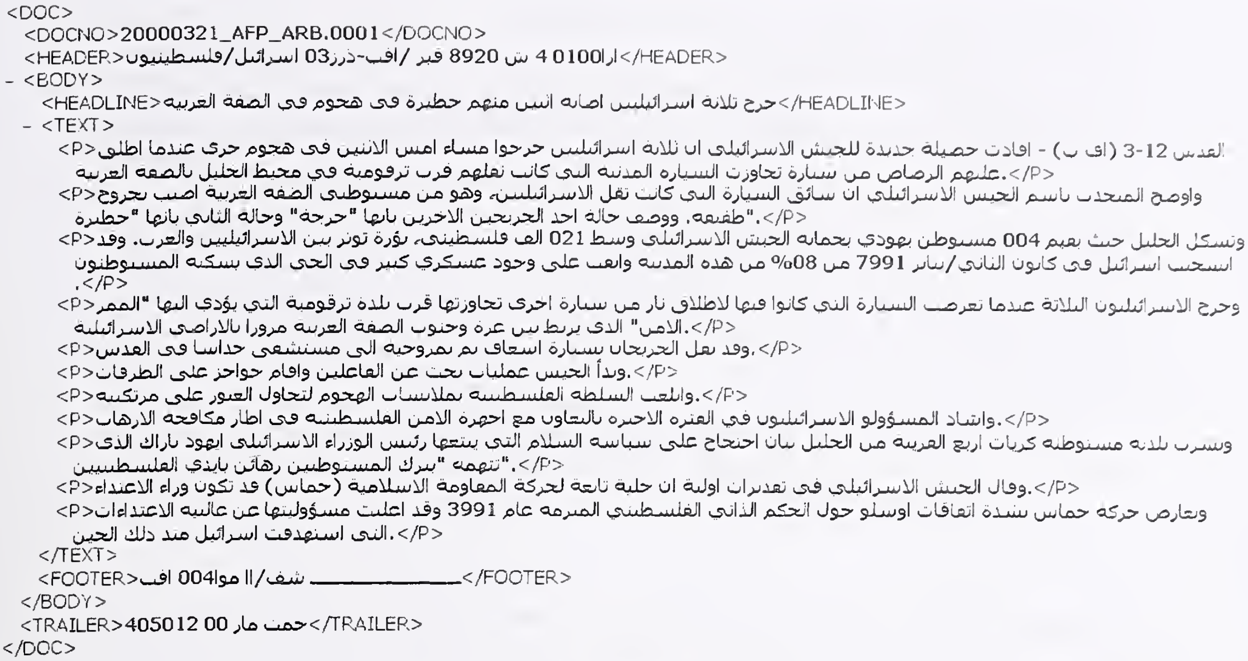
```
<top>
<num>Number: AR26</num>
<title>Kurdistan Independence</title>
<desc> Description:
How does the National Council of Resistance relate to the potential
independence of Kurdistan? </desc>

<narr> Narrative:

Articles reporting activities of the National Council of Resistance
are considered on topic. Articles discussing Ocalan's leadership
within the context of the Kurdish efforts toward independence are
also considered on topic.</narr>

</top>
```

The Linguistic Data Consortium also prepared an Arabic translation of the topics, so participating teams also had the option of doing monolingual (Arabic-Arabic) retrieval. The same topic in Arabic was distributed as:

```
<top>
<num> Number: AR26 </num>
<title> مجلس المقاومة الوطني الكردستاني </title>
<desc> Description:

كيف ينظر مجلس المقاومة الوطنية الى الإستقلال المحتمل للاكراد؟

</desc>
<narr> Narrative:
. الموضوع يتضمن نصوص متعلقة بتحركات مجلس المقاومة الوطنية ، مقالات تتحدث عن قيادة اوجلان ضمن جهود الاكراد للاستقلال

</narr>
</top>
```

2.2 Documents

As in the TREC-2001 CLIR track, the document collection contained 383,872 newswire stories that appeared on the Agence Française de Presse (AFP) Arabic Newswire between 1994 and 2000. The documents were represented in Unicode and encoded in UTF-8, resulting in an 896 MB collection. A typical document is shown in Figure 1.



```
<DOC>
  <DOCNO>20000321_AFP_ARB.0001</DOCNO>
  <HEADER>ارا0100 4 ش 8920 قبر /افب-ذرز03 اسرائيل/فلسطينيون</HEADER>
- <BODY>
    <HEADLINE>جرح ثلاثة اسرائيليين اصابه اثنين منهم خطيرة في هجوم في الصفة العربيه</HEADLINE>
  - <TEXT>
      <P>العدس 3-12 (اف ب) - افادت حصيلة جديدة للجيش الاسرائيلي ان ثلاثة اسرائيليين جرحوا مساء امس الاثنين في هجوم جرى عندما اطلق
        عليهم الرصاص من سيارة تجاوزت السياره المدنيه التي كانت تقلهم قرب ترقومية في محيط الخليل بالصفة العربية.</P>
      <P>واوضح المتحدث باسم الجيش الاسرائيلي ان سائق السيارة التي كانت تقل الاسرائيليين، وهو من مستوطني الضفه العربية اصيب بجروح
        خطيرة "طفيفه. ووصف حالة احد الجريحين الاخرين بانها "جرحة" وحالة الثاني بانها".</P>
      <P>وتسكل الخليل حيث يقيم 004 مستوطن يهودي بحمايه الجيش الاسرائيلي وسط 021 الف فلسطيني، بؤرة توتر بين الاسرائيليين والعرب. وقد
        انسحب اسرائيل في كانون الثاني/ينار 7991 من 08% من هذه المدينه وابقت على وجود عسكري كبير في الحي الذي يسكنه المستوطنون
        .</P>
      <P>وجرح الاسرائيليون الثلاثة عندما تعرضت السيارة التي كانوا فيها لاطلاق نار من سيارة اخرى تجاوزتها قرب بلدة ترقومية التي يؤدي اليها "الممر
        الامن" الذي يربط بين غزة وجنوب الصفة العربيه مرورا بالاراضي الاسرائيلية.</P>
      <P>وقد نقل الجريحان بسيارة اسعاف تم بمروحيه الى مستشفى هداسا في العدس.</P>
      <P>وبدأ الجيش عمليات بحث عن الفاعلين واقام حواجز على الطرقات.</P>
      <P>وابلغت السلطه الفلسطينيه بملابسات الهجوم لتحاول العثور على مرتكبيه.</P>
      <P>واشاد المسؤولو الاسرائيليون في الفتره الاخيره بالتعاون مع اجهزة الامن الفلسطينيه في اطار مكافحه الارهاب.</P>
      <P>وتسرب بلدنه مستوطنه كريات اربع العربيه من الخليل بان احتجاج على سياسه السلام التي ينتهجها رئيس الوزراء الاسرائيلي ايهود باراك الذي
        "تتهمه "بترك المستوطنين رهائن بايدي الفلسطينيين.</P>
      <P>وقال الجيش الاسرائيلي في تقديرات اولية ان خلية تابعة لحركة المقاومة الاسلامية (حماس) قد تكون وراء الاعتداء.</P>
      <P>وتعارض حركة حماس بشدة اتفاقات اوسلو حول الحكم الذاتي الفلسطيني المبرمة عام 3991 وقد اعلنت مسؤوليتها عن غالبية الاعتداءات
        التي استهدفت اسرائيل منذ ذلك الحين.</P>
    </TEXT>
    <FOOTER>شف/اا موا004 افب ______________</FOOTER>
  </BODY>
  <TRAILER>حمت مار 00 405012</TRAILER>
</DOC>
```

Figure 1. A sample Arabic document from the AFP collection.

3 Relevance Judgments

The nine participating teams shown in Table 1 together produced 23 automatic cross-language runs, 17 automatic monolingual runs with Arabic queries, and one manual monolingual run. The total number of relevant documents found over 50 topics was 5,909 with a mean of 118 relevant documents per topic, a maximum of 523, and minimum of 10. In TREC-2001, such a large number of relevant documents were uniquely found by individual runs that there was some concern about the comparability of post-hoc and official runs. In the pooled relevance judgment methodology, most documents remain unjudged, and the usual procedure is to treat unjudged documents as if they are not relevant. Voorhees has shown that the preference order between automatic runs in the TREC ad hoc retrieval task would rarely be reversed by the addition of missing judgments, and that the relative reduction in mean uninterpolated average precision that would result from removing "uniques" (relevant documents found by only a single system) from the judgment pools is typically less than 5% [2]. In TREC-2001 CLIR collection, 9 of 28 judged automatic runs experienced a relative reduction in mean uninterpolated average precision exceeding 10% relative when "uniques" contributed by that run were removed from the judgment pool. As Figure 2 shows, for the TREC-2002 CLIR collection, only one of 41 judged runs would experience more than 5% reduction. Figure 3 shows the unique relevant documents contributed to the final relevance pool by each group; no team uniquely contributed more than 6% of the known relevant documents (with the largest contribution coming from the one team that submitted a manual run). These results suggest that post-hoc use of the TREC-2002 collection will result in meaningful comparisons with the set of judged runs reported in this proceedings.

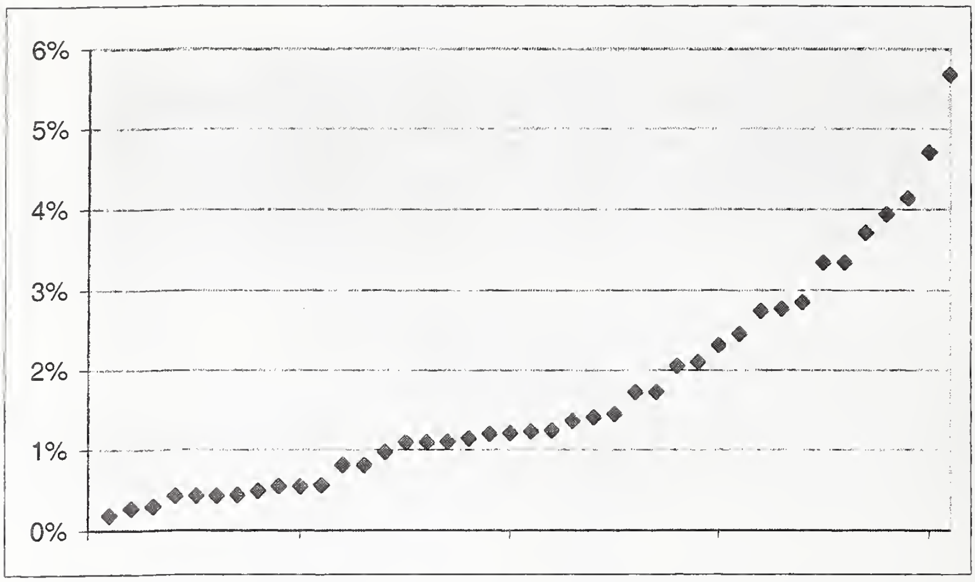


Figure 2. Effect of removing relevant documents found uniquely by each of 41 official runs.

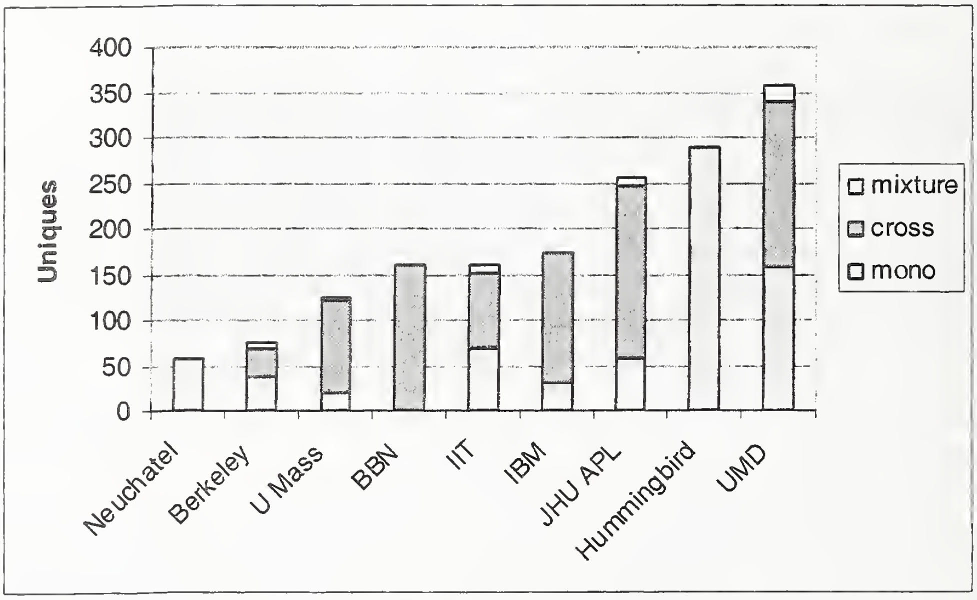


Figure 3. Unique contributions to known relevant documents, by participating team.

4 Standard Resources

Cross-language retrieval effectiveness depends on both the design of the retrieval system and the quality of the linguistic resources that are used. In order to begin to tease apart these factors, each participating team that performed the CLIR task was invited to submit one title+description run in which standard linguistic resources were used to the extent practical given their design. For example, a team using dictionary-based query translation could submit one run in which a standard bilingual dictionary was the only dictionary used. The following standard resources were made available:

- An Arabic "light" stemmer that used truncation rules to removed a small set of prefixes and suffixes. The light stemmer was developed through collaboration between Kareem

Darwish at the University of Maryland and Leah Larkey at the University of Massachusetts.

- A bidirectional Arabic-English bilingual dictionary, rekeyed from Salmone's Advanced Learner's Arabic-English Dictionary. The dictionary was provided by David Smith, of Tufts University.
- Two tables of translation probabilities, one for English-to-Arabic and the other for Arabic-to-English. These tables were developed using the Giza++ implementation of IBM Model 1 to align Arabic stems produced the track's standard light stemmer with English stems produced using the Porter stemmer. The documents on which this alignment was performed were obtained from the United Nations by Jinxi Xu at BBN and the alignments were produced by Alex Fraser of USC-ISI while working at BBN.
- A Web-based bidirectional Arabic-English machine translation system. We designated the system available at http://tarjim.ajeeb.com as the standard resource for the track.

Comparing the standard resource runs with the best runs by site (for the same query length) suggests that these resources were generally near, but not at, the state-of-the-art.

5 Retrieval Approaches

The nine groups have written papers about their methods and experiments. Three themes emerge across the reported work: (1) A greater focus on exploring innovative CLIR techniques than was evident in TREC-2001, (2) continued investigation of Arabic-specific issues, such as stemming and stopword removal, and (3) increasing reliance on multiple sources of evidence to overcome the limitation of any single source.

5.1 BBN

BBN made use of its probabilistic translation and retrieval model used in TREC-9 (for Chinese) and TREC-2002 (for Arabic) as well as the United Nations corpus. They employed the method of Hiemstra and de Jong to compute English probabilities by projecting Arabic terms to English (a weighted sum of corpus probabilities of Arabic terms where the weights are translation probabilities). To process the UN corpus and generate a new bilingual translation lexicon, BBN utilized the IBM model 4 statistical translation approach, rather than IBM model 1. For stemming they made use of the University of Massachusetts (Light8) stemmer in addition to the Buckwalter stemmer used in TREC-2001. In the area of query expansion they performed English and Arabic query expansions independently rather than sequentially (English query expansion followed by Arabic query expansion) as in TREC-2001.

5.2 University of California at Berkeley

The main focus of Berkeley's work was to develop several approaches to Arabic stemming and stopword list generation. Berkeley used the Ajeeb machine translation system to translate every word in the AFP collection after minimal word normalization. A 3,447-entry Arabic stopword list was created as the set of all Arabic terms that translated to an English stopword. A similar approach was used to generate a stemmer – Arabic words were partitioned into clusters based on their English translations, with Arabic words whose English translations were conflated to the same English stem forming a cluster. A second stemmer in which common one, two, and three character prefixes and suffixes (identified in the AFP corpus) were removed was also tried. These resources were then used in various combinations to perform dictionary-based retrieval using an extension of the logistic regression technique from previous TREC evaluations that incorporated blind relevance feedback. Berkeley also submitted a merged run in which two machine translation systems (Ajeeb and Al-Misbar) were used to perform query translation directly and the Salmone dictionary was used to perform dictionary-based query translation. Each was Score-based merging was then used to produce the final ranking.

5.3 Hummingbird Technologies

Hummingbird Technologies chose to focus on monolingual Arabic retrieval again this year. Hummingbird used a minimalist approach, with the same set of stemming rules as last year. Hummingbird makes a unique contribution to the CLIR track by evaluating commercially available technology that has been integrated into a comprehensive document management system.

5.4 Illinois Institute of Technology

In their TREC-2002 experiments with Arabic CLIR, Illinois Institute of Technology (IIT) continued their investigations of improvement of monolingual Arabic retrieval using different stemming approaches. For TREC-2001 IIT developed a 'light stemming' approach that performed well. For TREC-2002, they developed two new stemmers, one rule-based and one based on pattern matching. Both stemmers used an external training corpus from 1999-2001 editions of two Saudi Arabian newspapers to identify frequent prefixes and suffixes. Both stemmers performed comparably in monolingual Arabic retrieval, and the paper contains detailed examples of how each approach to stemming can affect word meaning. For CLIR, IIT used the Ajeeb machine translation package to translate the English queries into Arabic, and they also tried a second approach based on the translation probability tables provided as part of the standard resource set. In this case, MT produced better results, perhaps because the IIT stemmers differed from those used to produce the translation probability tables.

5.5 IBM Research

IBM's cross-language experiments were based entirely upon statistical machine translation using the IBM model 1 approach. IBM used the UN parallel corpus provided by BBN with a newly developed Arabic morphological analyzer. Two statistical machine translation systems were built. The first, an Arabic-to-English sentence translation model, was used to translate the documents into English, followed by monolingual English retrieval. The second, a probabilistic convolution model in which the probability of generating an English query stem was modeled based on the probabilities of generating that stem from an Arabic word or morpheme observed in the document. The convolution model substantially outperformed the sentence translation model, perhaps because it made use of document-wide translation probabilities.

5.6 Johns Hopkins University – Applied Physics Laboratory (JHU-APL)

JHU-APL continued its investigation of the use of overlapping character n-grams for monolingual Arabic retrieval. In TREC-2001 they used 4-grams; for TREC-2002 they investigated the use of 3-grams, 4-grams and 5-grams, and their official monolingual runs used combinations of those n-gram lengths. They used the same two machine translation systems as the Berkeley team, performing query translation from English to Arabic. The use of stemming in the standard translation probability tables made that resource unsuitable for straightforward use with n-gram-based techniques. They did, however, try mapping all English words that share a common stem to all Arabic words that share a common stem.

5.7 University of Maryland

The main focus of this year's experiments at Maryland was on combination-of-evidence techniques for cross-language retrieval. Translation knowledge was obtained from the same two machine translation systems that Berkeley used, the Salmone bilingual dictionary, and both directions of the BBN translation probability tables. Translation probabilities were estimated based on all of this evidence and then used with five CLIR techniques, one of which was a previously developed baseline (Pirkola's method). Some side experiments were also done to investigate the potential of document expansion and variants of light stemming.

5.8 University of Massachusetts

University of Massachusetts tried the most extensive set of techniques. Generally, acronyms found in the query were expanded using the U Mass *Acrophile* system. A bilingual lexicon built from a proper name dictionary from New Mexico State University and the same two MT systems that Berkeley used was then used to translate expanded English query terms and add them to the lexicon. Query expansion was performed both before and after translation. Arabic stopword removal was performed after morphological normalization using a 168-term stopword list from University of Lancaster, and several alternative forms of morphological normalization were tried. A number of variants on the processing path and the retrieval model were tried, and score-based merging among these variants was then used to create the final submissions.

5.9 University of Neuchatel

University of Neuchatel only submitted monolingual Arabic runs to the TREC-2002 CLIR Track evaluation. Neuchatel took the interesting approach of independently indexing Arabic words and of indexing tri-grams as alternative indexing and retrieval scheme, following the approach of Darwish and Oard in their SIGIR-2002 paper. Prior to all indexing and retrieval, Neuchatel converted and normalized the Arabic document text into Latin letters ("In Malta, the Arabic language is written using the Latin alphabet"). The Neuchatel word approach developed two stemmers which fall into the same area of 'light stemming' of the standard resource. They then dropped all stopwords which appeared in their 347 Arabic stopword list. From the paper it appears that their tri-gram approach utilized words as a basis (rather than including white space as other n-gram approaches usually do) and removed frequent tri-grams from a tri-gram stoplist generated from the stopword list above. Following independent retrieval of word-based indexed documents and tri-gram-based indexed documents, a "data fusion" summing approach to independently generated RSV rankings was applied to generate the final ranked list. They utilized the Rocchio approach to blind feedback and performed multiple relevance-expansion experiments on both stems and tri-grams.

6 Results

Monolingual runs establish a useful baseline to which cross-language results can be compared, and they help to enrich the relevance judgment pools. No standard query length was required for monolingual runs, but seven of the eight teams submitting monolingual runs elected to use title+description queries for at least one monolingual run. Figure 4 shows the best title+description monolingual run for those seven teams. The best of these results is somewhat below the best that was achieved last year. Since the document collection is the same, this suggests that the topics this year may be somewhat harder on average than last year's topics were.

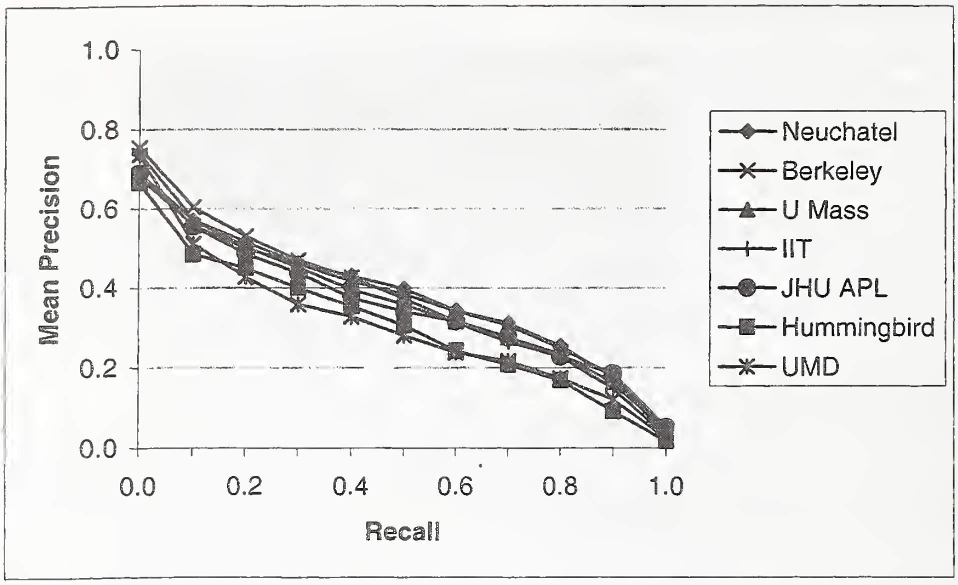


Figure 4. Best automatic monolingual runs, title+description queries.

Figure 5 shows the best automatic cross-language run for the required title+description condition for the seven teams that submitted at least one cross-language run. Five teams ran title+description queries for both the monolingual and the cross-language conditions, with the cross-language retrieval exhibiting a relative effectiveness ranging from 29% relative below monolingual to 6% relative above monolingual. It is difficult to draw any conclusions in the face of such large variability; interpretation of these results will require close attention to the implementation details of individual systems.

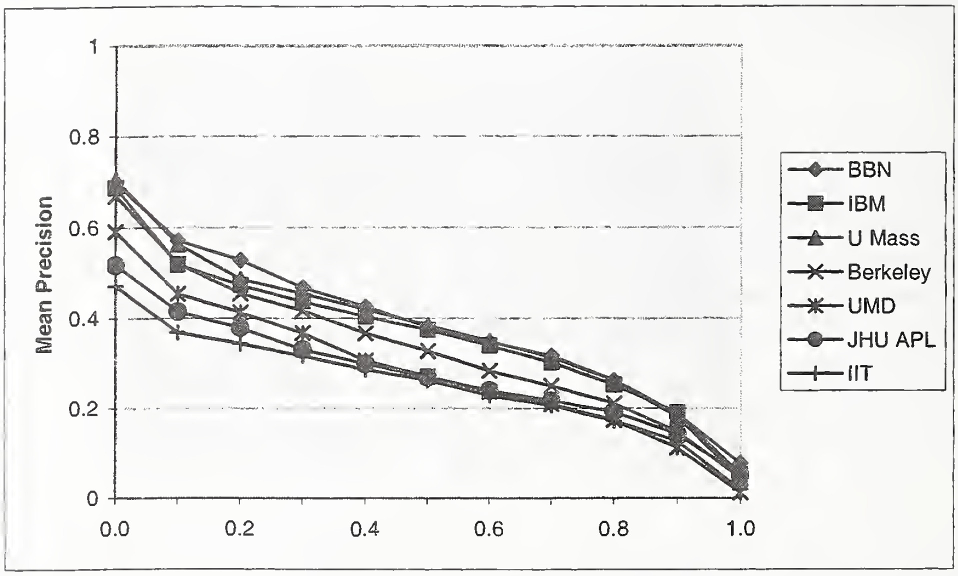


Figure 5. Best automatic cross-language runs, title+description queries.

Five teams submitted standard resource runs. As Figure 6 shows, three teams demonstrated improved performance (ranging from 4% to 11% relative) through the use of additional linguistic resources. Only five runs could be scored officially for each participating team, so post-hoc analysis may reveal greater potential for improvement from the use of additional linguistic resources than can be seen in the official results.

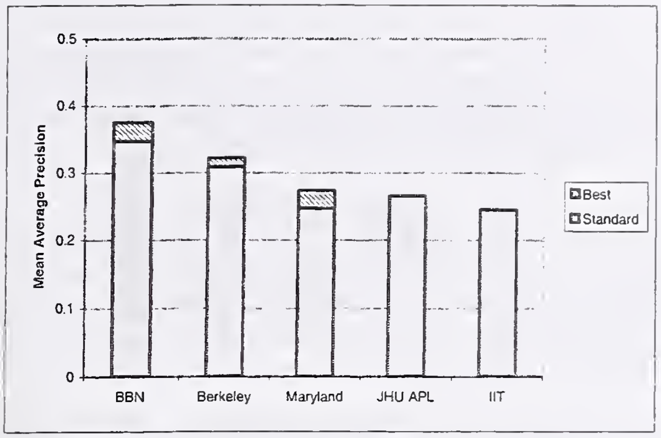


Figure 6. Improvement obtained using additional linguistic resources.

As is common in information retrieval evaluations, substantial variation was observed in retrieval effectiveness on a topic-by-topic basis. Figure 7, which plots the median and maximum average precision over the 23 cross-language runs illustrates this (note, however, that this plot includes queries of different lengths). For example, half of the runs did poorly on topics 34 and 55, but at least one run exceeded the median average precision for those topics by 800% relative. Applying this type of analysis on a run-by-run basis can help to identify effects that would be masked by averages across topics.

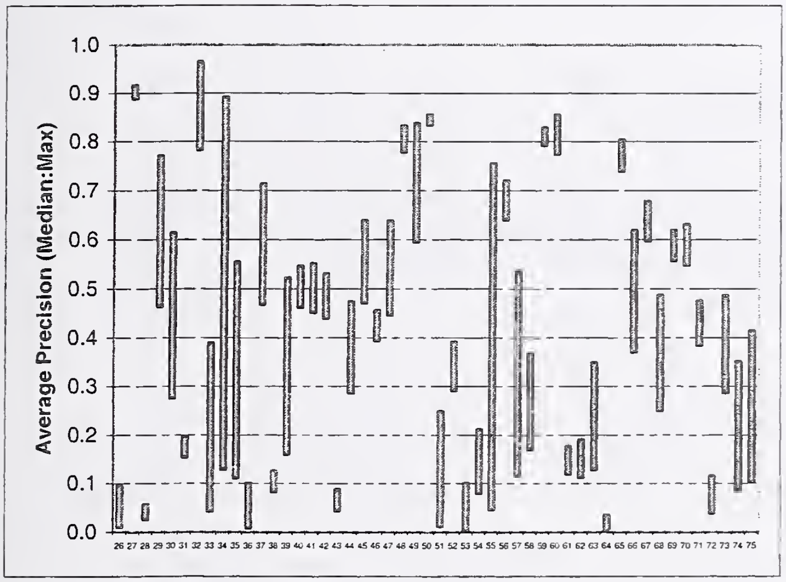


Figure 7. Average precision by topic (bottom=median, top=maximum), cross-language.

6 Looking to the Future

The TREC evaluations produce three things of enduring value: (1) research results, (2) standard test collections on which new techniques can be evaluated, and (3) a research community with shared interests. Over the past nine years, TREC has produced eight non-English test collections in six languages (Arabic, Chinese, French, German, Italian, and Spanish), and the demonstrated utility of these collections has inspired the creation of similar collections for many other languages (including Dutch, Japanese, Korean and Finnish). The results obtained this year indicate that topics AR26-AR75 are suitable for post-hoc use of the collection by automatic systems that did not contribute to the relevance pools. Topics AR1-AR25 have proven to be of some use for system tuning, but the relatively long title fields and the markedly elevated "uniques" effect make use of that collection for comparative studies less advisable. We therefore recommend that researchers working with this collection in the future report results for the 50 topics developed this year rather than treating all 75 topics as a single collection.

Our community now includes hundreds of researchers working on CLIR in dozens of countries, and research results regularly appear in a broad array of venues. Although this is the last year of the CLIR track at TREC, similar evaluations will continue in Europe at CLEF [3] and Japan at NTCIR [4], and work on searching Arabic will continue in the Topic Detection and Tracking evaluations (TDT) [5]. The idea of providing standard resources was first tried at TDT, and we have found it to be useful in a more traditional CLIR evaluation design as well.

Some of the questions that we have explored in the CLIR track may ultimately migrate into other tracks at TREC. CLIR is, after all, just one capability among the many that are needed to build effective systems to access globally distributed information. Perhaps the future will see the creation of a multilingual Web track, a track for searching multilingual speech, or an interactive multilingual track. When that happens, the baseline technology from which those specialized applications will be built will have been first developed here, in the TREC CLIR track.

Acknowledgments

The authors are grateful to Ellen Voorhees for handling all of the logistics for this track at NIST and for providing the data that was the basis for our analysis, to the LDC for their work on topic development and relevance assessment, to the contributors of the standard resources for making that comparison possible, and to the participating research teams for their advice and insights along the way. This work was supported in part by DARPA cooperative agreements N660010028910 and N660010018911.

References

[1] Fredric C. Gey and Douglas W. Oard. The TREC-2001 cross-language information retrieval track. In: *Proceedings of the 2001 Text Retrieval Conference*, NIST, 2001.

The TREC 2002 Filtering Track Report

Stephen Robertson
Microsoft Research
Cambridge, UK
ser@microsoft.com

Ian Soboroff
NIST
Gaithersburg, MD, USA
ian.soboroff@nist.gov

Abstract

The TREC–11 filtering track measures the ability of systems to build persistent user profiles which successfully separate relevant and non-relevant documents in an incoming stream. It consists of three major subtasks: adaptive filtering, batch filtering, and routing. In adaptive filtering, the system begins with only a topic statement and a small number of positive examples, and must learn a better profile from on-line feedback. Batch filtering and routing are more traditional machine learning tasks where the system begins with a large sample of evaluated training documents. This report describes the track, presents some evaluation results, and provides a general commentary on lessons learned from this year's track.

1 Introduction

A text filtering system sifts through a stream of incoming information to find documents relevant to a set of user needs represented by profiles. Unlike the traditional search query, user profiles are persistent, and tend to reflect a long term information need. With user feedback, the system can learn a better profile, and improve its performance over time. The TREC filtering track tries to simulate on-line time-critical text filtering applications, where the value of a document decays rapidly with time. This means that potentially relevant documents must be presented immediately to the user. There is no time to accumulate and rank a set of documents. Evaluation is based only on the quality of the retrieved set.

Filtering differs from search in that documents arrive sequentially over time. The TREC filtering track consists of three subtasks: adaptive filtering, batch filtering, and routing. In adaptive filtering, the system starts with only a user profile and a very small number of positive examples (relevant documents). It must begin filtering documents without any other prior information. Each retrieved document is immediately judged for relevance, and this information can be used by the system to adaptively update the filtering profile. In batch filtering and routing, the system starts with a larger set of evaluated training documents which can be used to help construct the search profile. For batch filtering, the system must decide to accept or reject each document, while routing systems can return a ranked list of documents. The core tasks for TREC–11 are very similar to those investigated in TREC–7 through TREC–10.

Traditional ad hoc retrieval and routing simulate a non-interactive process where users look at documents once at the end of system processing. This allows for ranking or clustering of the retrieved set. The filtering model is based on the assumption that users examine documents periodically over time. The actual frequency of user interaction is unknown and task-dependent. Rather than create a complex simulation which includes partial batching and ranking of the document set, we make the simplifying assumption that users want to be notified about interesting documents as soon as they arrive. Therefore, a decision must be made about each document without reference to future documents, and the retrieved set is ordered by time, not estimated likelihood of relevance. The history and development of the TREC Filtering Track can be traced by reading the yearly final reports:

- TREC–10 http://trec.nist.gov/pubs/trec10/t10_proceedings.html (#3) [9]
- TREC–9 http://trec.nist.gov/pubs/trec9/t9_proceedings.html (#3) [8]
- TREC–8 http://trec.nist.gov/pubs/trec8/t8_proceedings.html (#3 - 2 files) [4]
- TREC–7 http://trec.nist.gov/pubs/trec7/t7_proceedings.html (#3 - 2 files) [3]
- TREC–6 http://trec.nist.gov/pubs/trec6/t6_proceedings.html (#4 and #5) [2]
- TREC–5 http://trec.nist.gov/pubs/trec5/t5_proceedings.html (#5) [6]
- TREC–4 http://trec.nist.gov/pubs/trec4/t4_proceedings.html (#11) [5]

Information on the participating groups and their filtering systems can be found in the individual site reports, also available from the TREC web site.

2 TREC–11 Task Description

For those familiar with previous TRECs, the basic filtering tasks in TREC–11 are similar to those investigated in TREC–7 through TREC–10. The corpus is the same as for TREC–10, but a new set of topics has been prepared. In this section, we review the corpus, the three sub-tasks, the submission requirements, and the evaluation measures. For more background and motivation, please consult the TREC–7 track report [3].

2.1 Data

This year, the track has again used the RCV1 corpus provided by Reuters for research purposes [7]. This is a collection of about 800,000 news stories, covering a time period of a year in 1996-7. Items in the collection have unique identifiers and are dated but not timed. For the purpose of the experiment, it is assumed that the time-order of items within one day is the same as identifier order. (Item id on its own is insufficient for ordering, as there is some conflict across days). The first 6 weeks' items, 20 August through 30 September 1996, were taken as the training set (which could be used in ways specified below). The remainder of the collection formed the test set.

A new set of 100 topics was prepared for this year. Fifty of these were constructed in the traditional TREC fashion, by the assessors at NIST. In order to provide the necessary relevance judgements for training (including adaptive filtering), extensive searches using multiple retrieval and classification systems were conducted at NIST after initial definition of the topics, and the assessors made relevance judgements on the fused output. This process included several feedback stages, so that after one round of such assessment, relevance information was used to improve the queries and another round of assessments was made. Feedback continued until no more relevant documents were found in a given round, or until five rounds had passed. Each topic received between two and seven rounds of judging (some topics had more than five rounds due to glitches in the feedback system).

Additional relevance judgements were made for these assessor topics after submission of results by the participants, on documents taken from the pooled submissions for each topic. These resulted in the identification of additional relevant documents, which were not available to the adaptive systems, but which were included for the purpose of evaluating all systems. All results below are based on the full set of relevance judgements. Further details and analysis on this post-submission phase of judgements is given below (section 4.1). Additional discussion of both pre- and post-submission judgements, and the whole process of constructing the new topic sets, is given in [10].

The remaining fifty topics were constructed as intersections of pairs of Reuters categories. Pairs of categories were chosen to be apparently meaningful as search topics, to have a minimum of three relevant documents in the training set, and to have an overall number of relevant documents in the range of the assessor-built topics. (Relevant documents are here defined as documents assigned both category labels in the Reuters collection.) For the purposes of training for batch filtering and routing, and in order to make this set of topics similar to the previous set of 50, a selection of non-relevant documents was included in the set of relevance judgements provided for each topic. These non-relevant documents were chosen randomly from those assigned either of the category labels, but not both. This places the non-relevant documents in the "neighbourhood" of the intersection, hopefully similar to highly-ranked documents in a pool which are judged irrelevant by an assessor.

This second set of topics represents a trial of a relatively cheap way of constructing topics for retrieval experiments, given a collection with category labels assigned. It is regarded as an experiment to assess whether such a methodology is likely to be useful for future experiments.

2.2 Tasks

The adaptive filtering task is designed to model the text filtering process from the moment of profile construction. In TREC–11, following the idea first used in TREC–9, we model the situation where the user arrives with a small number of known positive examples (relevant documents). For each topic, the last three relevant documents in the training set were made available to the participants for this purpose; no other relevance judgements from the training set could be used. Subsequently, once a document is retrieved, the relevance assessment (when one exists) is immediately made available to the system. Unfortunately, it is not feasible in practice to have interactive human assessment by NIST. Instead, assessment is simulated by releasing the pre-existing relevance judgement for that document. Judgements for unretrieved documents are never revealed to the system. Once the system makes a decision about whether or not to retrieve a document, that decision is final. No back-tracking or temporary caching of documents is allowed. While not always realistic, this condition reduces the complexity of the task and makes it easier to compare performance between different systems.

Systems are allowed to use the whole of the training set of documents (but no other relevance judgements than the three provided for each topic) to generate collection frequency statistics (such as inverse document frequencies) or auxiliary data structures (such as automatically-generated thesauri). Resources outside the Reuters collection could also be used. As documents were processed, the text could be used to update term frequency statistics and auxiliary document structures even if the document was not matched to any profile. Groups had the option to treat unevaluated documents as not relevant.

In batch filtering, all the training set documents and all relevance judgements on that set are available in advance. Once the system is trained, the test set is processed in its entirety. For each topic, the system returns a single retrieved set. For routing, the training data is the same as for batch filtering, but in this case systems return a ranked list of the top 1000 retrieved documents from the test set. Batch filtering and routing are included in order to encourage participation to as many different groups as possible.

2.3 Evaluation and optimisation

For the TREC experiments, filtering systems are expected to make a binary decision to accept or reject a document for each profile. Therefore, the retrieved set consists of an unranked list of documents. This fact has implications for evaluation, in that it demands a measure of effectiveness which can be applied to such an unranked set. Many of the standard measures used in the evaluation of ranked retrieval (such as average precision) are not applicable. Furthermore, the choice of primary measure of performance will impact the systems in a way that does not happen in ranked retrieval. While good ranking algorithms seem to be relatively independent of the evaluation measure used, good classification algorithms need to relate very strongly to the measure it is desired to optimise.

Two measures were used in TREC–11 for this purpose (as alternative sub-tasks). One was essentially the linear utility measure used in previous TRECs, and described below. The other was a version of the van Rijsbergen measure of retrieval performance, first used in TREC–10.

F-beta

This measure, based on one defined by van Rijsbergen, is a function of recall and precision, together with a free parameter beta which determines the relative weighting of recall and precision. For any beta, the measure lies in the range zero (bad) to 1 (good). For TREC–11 (as for TREC–10), a value of beta=0.5 has been chosen, corresponding to an emphasis on precision (beta=1 is neutral). The measure (with this choice of beta) may be expressed as follows:

$$\mathrm{T11F} = \frac{1.25 \times \text{No. of relevant docs retrieved}}{\text{No. of retrieved docs} + 0.25 \times \text{No. of relevant docs}}$$

(T11F is defined as zero if the number of retrieved documents is zero.)

Linear utility

The idea of a linear utility measure has been described in previous TREC reports (e.g. [4]). The particular parameters being used are a credit of 2 for a relevant document retrieved and a debit of 1 for a non-relevant document retrieved:

$$\mathrm{T11U} = 2 \times \text{No. of relevant docs retrieved} - \text{No. of non} - \text{relevant docs retrieved}$$

which corresponds to the retrieval rule:

$$\text{retrieve if } \ P(\text{rel}) > .33$$

Filtering according to a linear utility function is equivalent to filtering by estimated probability of relevance; the corresponding probability threshold is shown.

When evaluation is based on utility, it is difficult to compare performance across topics. Simple averaging of the utility measure gives each retrieved document equal weight, which means that the average scores will be dominated by the topics with large retrieved sets (as in micro-averaging). Furthermore, the utility scale is effectively unbounded below but bounded above; a single very poor query might completely swamp any number of good queries.

For the purpose of averaging across topics, the method used for TREC–11 is a slightly modified version of one used in TREC–9 (modification proposed by Ault). First, utilities are normalised by the maximum possible utility for the topic, namely

$$\mathrm{MaxU} = 2 \times (\text{No. of relevant docs})$$

I.e.

$$\mathrm{T11NU} = \frac{\mathrm{T11U}}{\mathrm{MaxU}}$$

The lower limit is some negative normalised utility, MinNU, which may be thought of as the minimum (maximum negative) utility that a user would tolerate, over the lifetime of the profile. If the T11NU value falls below this minimum, it will be assumed that the user stops looking at documents, and therefore the minimum is used. For each topic,

$$\mathrm{T11SU} = \frac{\max(\mathrm{T11NU}, \mathrm{MinNU}) - \mathrm{MinNU}}{1 - \mathrm{MinNU}}$$

and MeanT11SU is the mean of T11SU over topics.

Different values of MinNU may be chosen. The primary evaluation measure has

$$\mathrm{MinNU} = -0.5$$

Other measures

In the official results tables, a number of measures are included as well as the measure for which any particular run was specifically optimised. The range is as follows:

For adaptive and batch filtering:

- Mean T11SU (scaled utility) over topics, over the whole period and broken down by time period for adaptive filtering. Note that this is referred to in the tables as T11U, but is in fact T11SU.
- Mean T11F (F-beta, with beta = 0.5) over topics.
- Mean set recall
- Mean set precision
- Zeros (number of topics for which no documents were retrieved over the period)

All means are macro-averages, that is, averaged across topics. For routing, the usual range of ranked-output performance measures computed by `trec_eval` are given.

2.4 Submission Requirements

Each participating group could submit a limited number of runs, in each category: Adaptive filtering 4; Batch filtering 2; Routing 2.

Any of the filtering runs could be optimised for either T11F or T11SU – a declaration was required of the measure for which each run was optimised. There were no required runs, but participants were encouraged to provide an adaptive filtering run with T11SU optimisation.

Groups were also asked to indicate whether they used other parts of the TREC collection, or other external sources, to build term collection statistics or other resources.

3 TREC–11 results

Twenty one groups participated in the TREC–11 filtering track (two more than in TREC–10) and submitted a total of 73 runs (seven more than in TREC–10)). These break down as follows: 14 groups submitted adaptive filtering runs, 10 submitted to batch filtering, and 10 to routing.

Here is a list of the participating groups, including abbreviations and run identifiers. Participants will generally be referred to by their abbreviations in this paper. The run identifiers can be used to recognise which runs belong to which groups in the plotted results.

	Abbreviation	Run identifier
University of North Texas	north_texas	UNTextCat
KerMIT Consortium	kerMIT	KerMIT
Carnegie Mellon University	cmu_lti	CMUDIR
University of Hertfordshire	hertfordshire	UHcl
Microsoft Research Cambridge	microsoft_cambridge	ok11, msPUM
Moscow Medical Academy	moscow_med	mma2002
Rutgers University	rutgers-kantor	dimacs11
David D. Lewis, Independent Consultant	Lewis	dimacsdd
SUNY Buffalo	buffalo_cedar	cedar02
CLIPS Laoratory, IMAG	clips-imag	relief
National Institute of Informatics	nii	kNII11
Clairvoyance Corporation	clairvoyance	CCT11
Institut de Recherche en Informatique de Toulouse	irit	iritsig
Tampere University of Technology	tampere	Visa
Fudan University	Fudan	FDUT11
Queens College, City University of New York	cuny	pirc2
Chinese Academy of Sciences	chinese_academy	ICT
Queensland University of Technology	queensland	QUT
Johns Hopkins University Applied Physics Lab	jhu_apl	apl11
Tsinghua University	tsinghua	thuT11
University of Iowa	uiowa	UIowa02

3.1 Summary of approaches

These brief summaries are intended only to point readers toward other work. Not all groups have a paper in the proceedings.

University of North Texas participated in the batch filtering and routing tasks. Their TextCat system employs multiple simple text classifiers (an n-gram based one and Ripper) which may be combined by stacking them in series or using a voting scheme.

KerMIT Consortium participated in all three tasks. Their focus is on support vector machine (SVM) kernel methods. For routing, they used a linear SVM, and for batch filtering used the same SVM with a threshold selection mechanism. For adaptive filtering, they used second-order perceptrons and combined SVMs and perceptrons with uneven margins.

CMU participated in the adaptive filtering task. Their system was the same as used in TREC 9 and 10 and uses Rocchio's algorithm for profile updating. Their thresholding and term selection processes were chosen and tuned using past TREC filtering data.

University of Hertfordshire participated in the routing task. They manually selected sets of keywords using the topic descriptions and the adaptive training examples.

Microsoft Research Cambridge participated in the adaptive filtering and routing tasks. Their probabilistic Okapi/Keenbow system is very similar to that used in previous years, but the adaptive filtering component was rewritten for this year. The new filtering component allows updating of profiles and thresholds at each document retrieved. For routing, a new system using perceptrons with uneven margins was used.

Rutgers University participated in the adaptive and batch filtering tasks.[1] Their adaptive system is based on a Rocchio classifier and pseudo-relevance feedback. For batch filtering, they used rank-based feature

[1] David Lewis, part of the Rutgers group, submitted runs two adaptive filtering runs as a separate group. His results are presented in the Rutgers proceedings paper.

selection to identify a very small set of features to represent the collection and trained a simple classifier using these features.

State University of New York at Buffalo participated in the adaptive and batch filtering tasks. They used two main approaches, SVMs with weighted margins and language modeling.

CLIPS participated in the adaptive filtering task. Their RELIEFS system, introduced in TREC 9, is based on a probabilistic model of terms and relevance. This year, they focused on threshold adaptation and estimating relevance.

National Institute of Informatics participated in the batch filtering task. Their approach involved reweighting terms co-occurring in relevant training documents, and modeling these term sets as "virtual" relevant documents. They then used SVMs to learn a decision boundary based on the enlarged training set.

Clairvoyance Corporation participated in the batch filtering task. Their experiments focused on the performance of the monolithic filters which in their CLARIT system can be arranged to create ensemble filters. Their paper describes post-TREC experiments comparing their IR-based approaches to SVMs.

IRIT participated in all three tasks. Their Mercure system is based on a connectionist model. This year their experiments focused on threshold calibration.

Tampere University of Technology participated in the routing task. Their approach is based on word coding and characterizing the histograms of encoded documents.

Fudan University participated in the adaptive filtering task. They used the topic and training samples to create an initial Winnow classifier, and with that gathered a larger set of pseudo-relevant documents to further train the classifier.

Queens College, CUNY participated in the adaptive filtering task. They used a two-stage approach; initially, a simple profile reweighting and threshold adjustment scheme is used; but as more relevance information is available, the profile is expanded.

Chinese Academy of Sciences participated in all three tasks. Their experiments in adaptive filtering focused on making use of retrieved documents whose relevance is unknown in profile adaptation.

JHU/APL participated in all three tasks. For filtering, they used linear SVMs, with system parameters tuned using the TREC-8 filtering data. For routing, one run used SVMs and the other run merged the SVM run with an unsubmitted language modeling-based run.

Tsinghua University participated in the adaptive filtering task. Their incremental learning approach uses pseudo-relevance feedback to form the initial profile and threshold. They also experimented with a language modeling run using the Lemur toolkit.

University of Iowa participated in the adaptive filtering task. Their system uses two-level dynamic clustering. Documents placed into a topics first-level cluster are further divided into secondary clusters which are responsible for determining whether a document will be retrieved.

3.2 Evaluation Results

Some results are presented in the following graphs. Figures 1 and 2 show the adaptive filtering results for the utility and F measures. In each graph, the horizontal line inside a run's box is the median topic score, the box shows interquartile distance, the whiskers extend to the furthest topic within 1.5 times the interquartile distance, and the circles are outliers. In all graphs of T11SU scores, the horizontal line through the graph shows the baseline utility which can be achieved by retrieving no documents.

Figures 3 and 4 show the utility and F-beta results for batch filtering. Figure 5 shows mean uninterpolated average precision for routing.

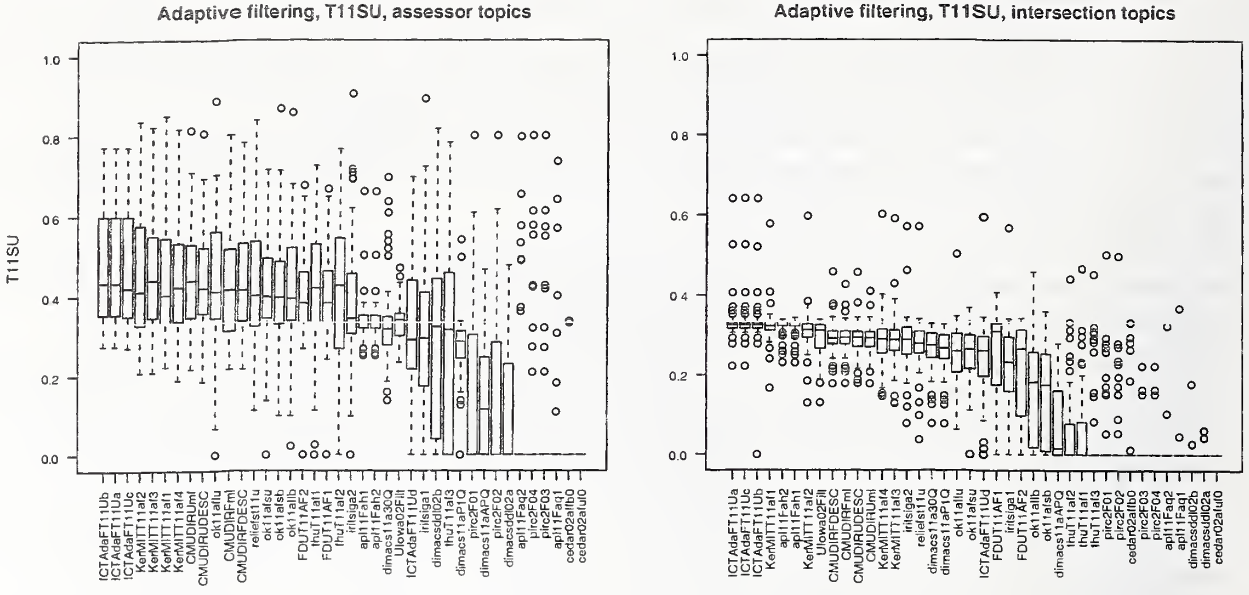


Figure 1: Adaptive filtering – T11SU

4 General Commentary

4.1 Post-submission judgements

Although more than 21,000 relevance judgements were made during topic creation and released with the topics, we were concerned that participants would still find more relevant documents. In order to make sure systems were measured fairly, NIST pooled participants' runs and judged any previously unjudged documents in the pool. Pooling was done as follows. Each participating group was allotted a fixed budget of documents to be pooled from their runs. If the group had any routing runs, we added unjudged documents from the top 100 ranks to the pool. If the group also had filtering runs, at most half the budget was expended on routing documents. We then merged all batch and adaptive filtering runs from that group and took a random sample of documents from the combined runs to fill out the pool budget. In all, another 42,000 documents were judged during this second round of assessment.

Figure 6 shows the numbers of relevant documents found for each topic in the first and second rounds of judging. Note that overall the topics have between 9 and 599 relevant documents apiece, much fewer than the TREC 2001 categories and closer to TREC ad hoc scale. For most topics only a few new relevant documents were found in the second round (median = 8.5), but seven topics had more than fifty new. Four of these topics had more than twenty new relevant documents found in their last round of feedback during the creation phase. Although our pooling process is radically different, these findings agree with Harman's analysis of the TREC-3 relevance judgements [1], as well as those of Zobel [11] that the "largest" topics (those with the most relevant documents) tend to yield even more relevant documents upon further searching. We have seen that such topics tend to have a greater number of relevant documents found in the last round of judging. In retrospect it probably would have been a good idea to discard these topics.

Another important factor is that five topics were judged by a different assessor in the second round than the one who had created it. Although as a general rule assessors always judged their own topics, due to time constraints we were forced to move these topics to different assessors. In these cases, the assessor was shown all of the relevant documents found in the first round as orientation to the topic. Four of these

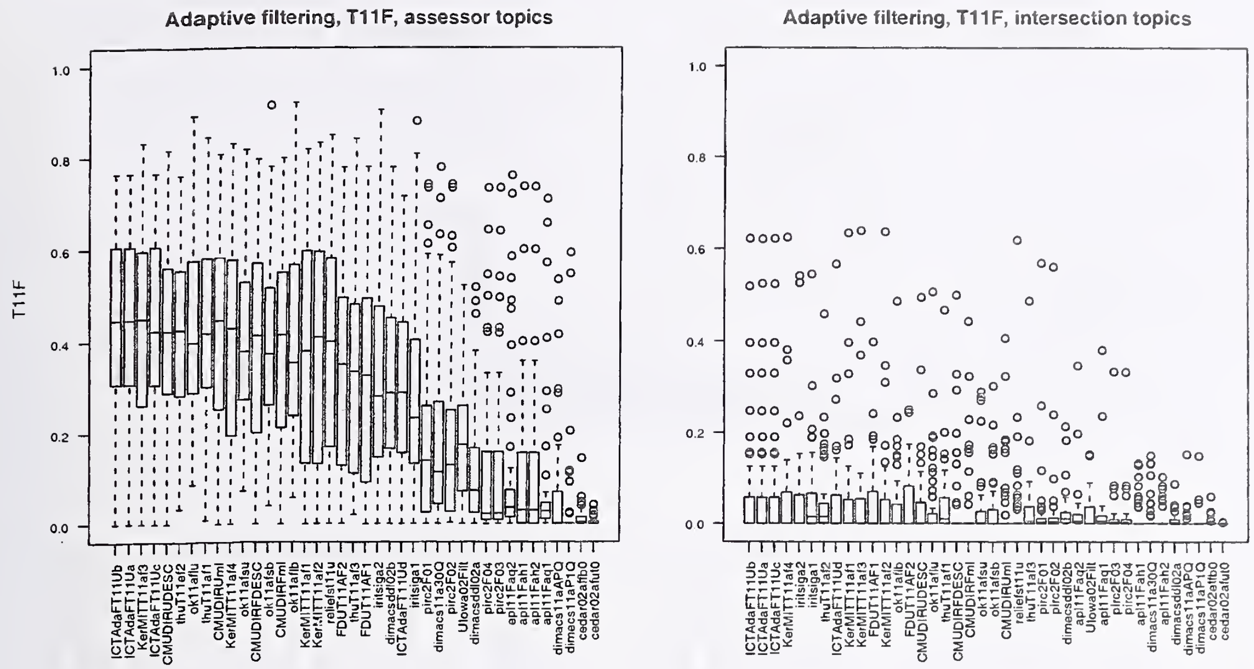


Figure 2: Adaptive filtering – T11F

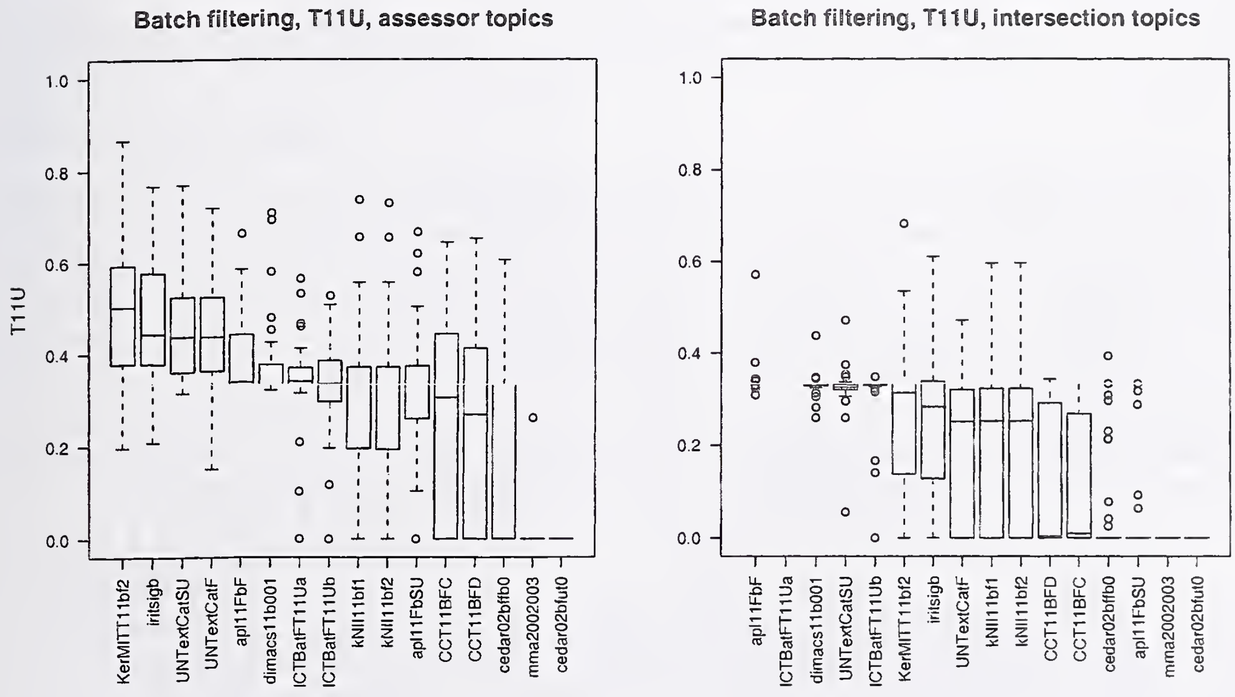


Figure 3: Batch filtering – T11SU

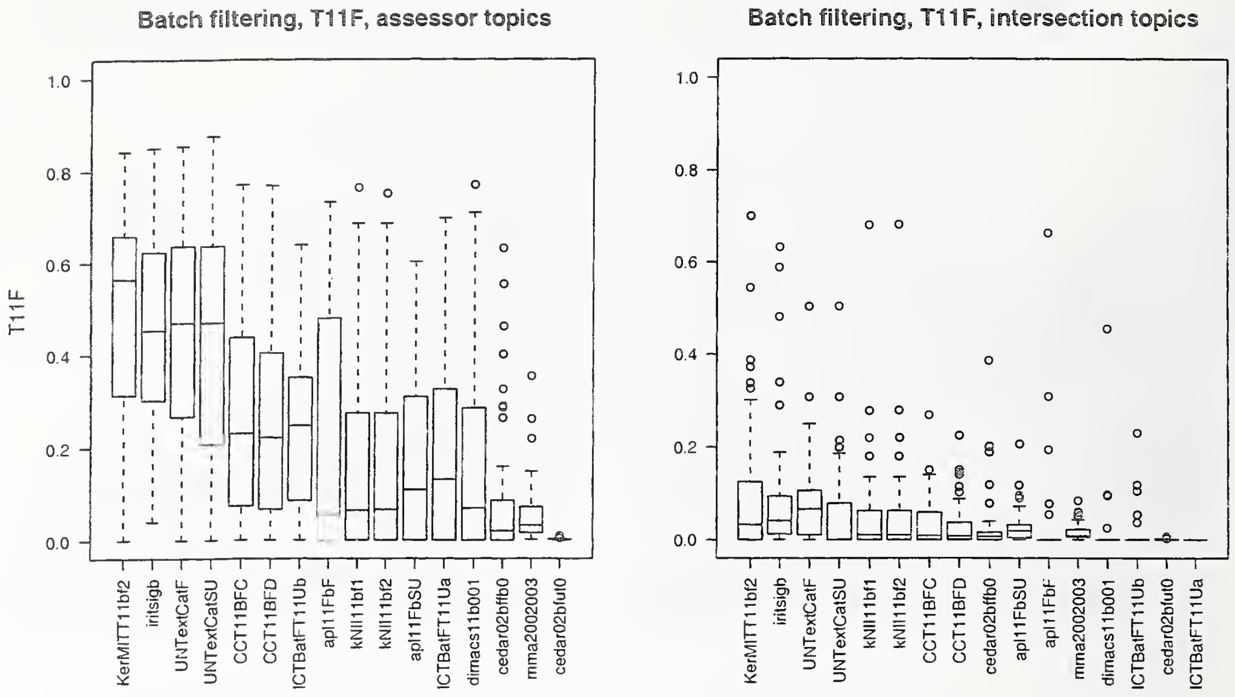


Figure 4: Batch filtering – T11F

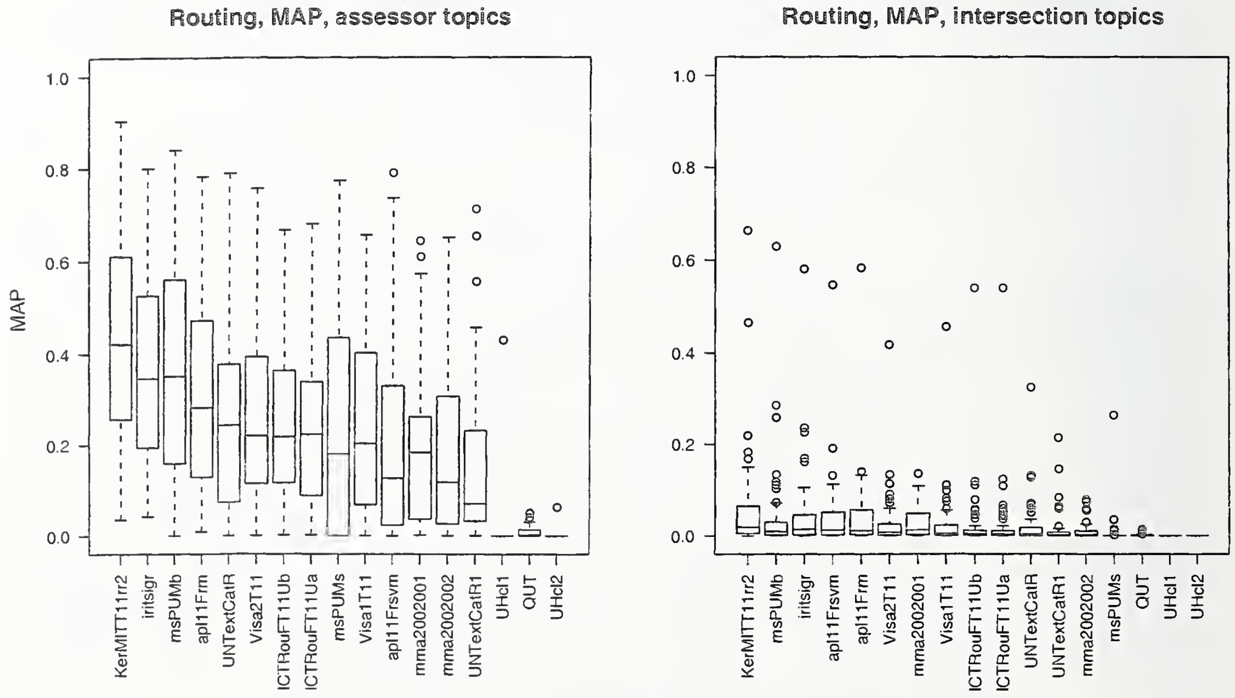


Figure 5: Routing – Mean Average Precision

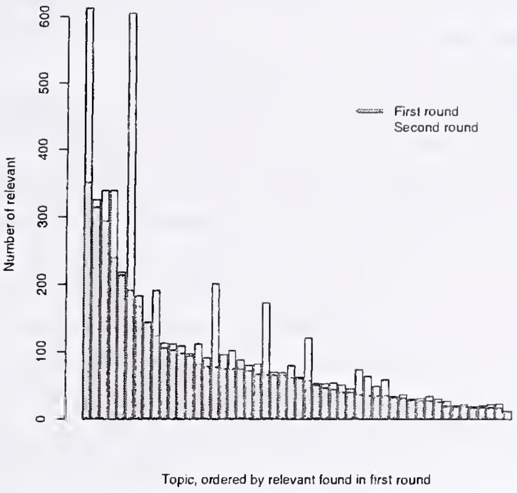


Figure 6: Relevant documents found in the first and second rounds of judging.

	T11U	T11F
Adaptive	0.969	0.936
Batch	0.996	0.983
Routing	0.912 (MAP)	

Table 1: Correlation of the official TREC results to a system ranking measured using the first-round relevance judgements only.

"moved" topics were also topics with more than fifty new relevant found, suggesting that these topics were not judged as well as the others.

4.2 Intersection topics

One issue this year concerned the experiment on the intersection method of building topics and making relevance judgements. This method would be considerably cheaper than the usual method involving assessors for both tasks: the process of making relevance judgements is a substantial effort. Our hope was that it would prove to be a viable alternative, providing a way of constructing test collections with much larger numbers of topics than we have at present, even if the quality is not quite as good.

In the event, the immediate impression of the intersection topics must be that they are not useful. The discrepancy in performance between assessor and intersection topics is huge. We might be tempted to hypothesise that the intersection topics are simply much harder than the assessor topics, but nevertheless represent a realistic task. However, it is hard to maintain that view in the light of the size of the discrepancy.

Possible hypotheses

Some hypotheses have been suggested (by participants and in subsequent discussion) for why the performance on the intersection topics was so poor. Roughly speaking, we may divide them into two classes: those which focus on the individual classes and those which focus on the intersection operation. In some cases at least, the hypothesis suggests experiments that may help to elucidate the problem; failure analysis on the official or other runs may also be informative. By and large these experiments and analyses have not yet been performed, although a few participants have done some failure analysis – they require some thought and effort, and being directed at a methodological question, they are not about specific systems, models or approaches, and therefore maybe of less immediate interest to participants. Nevertheless, the methodological question is of interest, and deserves investigation.

One hypothesis is that Reuters' assignment of category labels is simply too inconsistent, compared to assessor relevance judgements, to allow a system to learn adequately how to predict it. This hypothesis would suggest that the same problem would apply to topics defined as individual classes as to topics defined as intersections of pairs. However, the TREC 2001 experiment used individual classes and although many systems had significant difficulties, several systems performed adequately well on these topics. This would tend to suggest that the hypothesis as it stands is not a sufficient explanation.

A second is that Reuters' rules for category assignment specify that at least one category must be assigned to each document. Editors are happy if they can assign one category; extra ones are only assigned if (a) they immediately stand out as necessary, or (b) there is significant doubt about which is the correct one.[2] Either way, there is likely to be significantly more noise in second category assignment than in first category assignment, which will adversely affect the intersection topics. Experiments could be designed to substantiate this hypothesis.

A third is that categories may be of very different sizes; an intersection of a large category with a small one may be difficult to learn. A variant on this is qualitative rather than quantitative: some categories may be much harder to learn than others, and an intersection may be as hard as the harder of the two categories.

4.3 Overall performance

On the utility measure, most of the adaptive systems now outperform the baseline system which retrieves no documents ever. This is a welcome result. Furthermore, on the whole the adaptive systems are performing similarly to the batch filtering systems. In other words, despite starting from considerably less information they can through adaptation pull themselves up to a similar level overall. This suggests that at the end of the time period, they are likely to perform better than the batch systems.

Acknowledgements We give our thanks to all the people who have contributed to the development of the TREC filtering track over the years, in particular David Lewis, David Hull, Karen Sparck Jones, Chris Buckley, Paul Kantor, Ellen Voorhees, the TREC program committee, and the team at NIST.

References

[1] D K Harman. Overview of the Third Text REtrieval Conference (TREC–3). In D K Harman, editor, *Proceedings of the Third Text REtrieval Conference (TREC–3)*, pages 1–20. Gaithersburg, MD: NIST, 1994. NIST Special Publication 500-225.

[2] D A Hull. The TREC-6 filtering track: Description and analysis. In E M Voorhees and D K Harman, editors, *The Sixth Text REtrieval Conference (TREC–6)*, pages 45–68. Gaithersburg, MD: NIST, 1998. NIST Special Publication 500-240.

[3] D A Hull. The TREC-7 filtering track: Description and analysis. In E M Voorhees and D K Harman, editors, *The Seventh Text REtrieval Conference (TREC–7)*, pages 33–56. Gaithersburg, MD: NIST, 1999. NIST Special Publication 500-242.

[2] Suggested by David Lewis (private communication)

[4] D A Hull and S Robertson. The TREC-8 filtering track final report. In E M Voorhees and D K Harman, editors, *The Eighth Text REtrieval Conference (TREC–8)*, pages 35–56. Gaithersburg, MD: NIST, 2000. NIST Special Publication 500-246.

[5] D Lewis. The TREC–4 filtering track. In D K Harman, editor, *The Fourth Text REtrieval Conference (TREC–4)*, pages 165–180. Gaithersburg, MD: NIST, 1996. NIST Special Publication 500-236.

[6] D Lewis. The TREC-5 filtering track. In E M Voorhees and D K Harman, editors, *The Fifth Text REtrieval Conference (TREC–5)*, pages 75–96. Gaithersburg, MD: NIST, 1997. NIST Special Publication 500-238.

[7] Reuters corpus volume 1. http://about.reuters.com/researchandstandards/corpus/. Visited 26 September 2002.

[8] S Robertson and D A Hull. The TREC-9 filtering track final report. In E M Voorhees and D K Harman, editors, *The Ninth Text REtrieval Conference (TREC–9)*, pages 25–40. Gaithersburg, MD: NIST, 2001. NIST Special Publication 500-249.

[9] S Robertson and I Soboroff. The TREC 2001 filtering track report. In E M Voorhees and D K Harman, editors, *The Tenth Text REtrieval Conference, TREC 2001*, pages 26–37. Gaithersburg, MD: NIST, 2002. NIST Special Publication 500-250.

[10] I Soboroff and S Robertson. Building a Filtering Test Collection for TREC 2002. To appear in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '03)*.

[11] J Zobel. How Reliable are the Results of Large-Scale Retrieval Experiments? In W B Croft, A Moffat, C J van Rijsbergen, R Wilkinson, and J Zobel, editors, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98)*, pages 307–314. ACM Press: Melbourne, Australia, August 1998.

TREC 2002 Interactive Track Report

William Hersh, Oregon Health & Science University, Portland, OR, USA

For TREC 2002, the Interactive Track completed the two-year cycle of observational studies begun in TREC 2001 and followed now by more controlled laboratory experiments focusing on question answering using Web data. Six research groups participated this year.

Background

The Interactive Track was one of the first track's at TREC (dating back to TREC-3) and has always had a small but dedicated following. The track has accomplished much during its existence, including a special issue of *Information Processing and Management* (May/June, 2001) [1]. A variety of findings have emerged from experiments carried out by track participants:

- Presentation of documents matters to users, i.e., better surrogates help but clustering does not [2]
- Users do not utilize relevance feedback as we might think/hope [3]
- Results from "batch" studies do not necessarily apply to real-world searchers [4]

However, there have been limitations to the generalizability of the results obtained:

- Study sample sizes are small and conditions are artificial
- Searcher populations used might not be generalizable
- But the same apply to non-interactive studies, e.g.,
  - Is one search on many queries any better than multiple searches on the same query?
  - Some of the queries from batch experiments given to real users showed they were too easy

To help set the future direction of the track, a workshop was held at SIGIR 2000 [5]. It was decided at the workshop and subsequently that the track would move to a two-year cycle that would allow increased data collection to better formulate and study hypotheses. In particular, in the first year of the cycle, groups would perform observational studies that increased the realism of task and generated experimental hypotheses for the following year. The TREC 2001 Interactive Track was first year of observational studies, with hypothesis-driven experiments to be performed in TREC 2002.

Data for searching

The track used on open version of the .GOV Web collection created for the TREC 2002 Web Track for searching. The collection was "open" in the sense that some links to pages outside the collection were presented and could be followed. This meant that the collection was intermediate in its stability between the live Web used by the track last year and a completely fixed, closed version of the .GOV collection, which was desired but not available in time for experimentation.

The collection was used by most participating groups as indexed and searched by the Panoptic search engine. The cited version does stemming and the homepage-finding feature is turned off. Results could be obtained in XML format by sending a query via CGI, e.g.,

trec.panopticsearch.com/gov/padre-sw_xml.cgi?collection=gov&query=bush

and getting back a padre_results packet. Experiments did not need to be limited to the interface defined by actual HTML pages returned by the Panoptic engine. Help on the use of the Panoptic search engine was available.

Tasks

Eight searcher tasks, analogous to those used in TREC 2001, were generated and tested. They are listed below. All searchers were to be given at least 10 minutes on each task once the actual searching began, and participating groups had to report the results as of the end of the ten-minute period in their proceedings papers. Groups optionally were able to report additional results, e.g., after 5 minutes, 15 minutes, etc..

There were four general searching activities from upon which the eight actual tasks were modeled. The four activities were:

1. Looking for personal health information
2. Seeking guidance on US government laws, regulations, guidelines, policy
3. Making travel plans
4. Gathering material for a report on a given subject

The searcher tasks were formulated in one of the following ways:

1. Find any N short answers to a question, to which there are multiple answers of the same type.
2. Find any N websites that meet the need specified in the task statement

The eight tasks proper were:

1. You are traveling from the Netherlands, and want to bring some typical food products as gifts for your friends. What are three kinds of food products from the Netherlands that you are not allowed to bring into the US? [Government Regulation]
2. You are concerned with privacy issues related to electronic information and would like to know what laws have been passed by the US Congress regarding these issues. Identify three such laws. [Government Regulation]
3. A friend has a private well which is the family's only source of drinking water. Locate a US publication, which contains guidelines for the maintenance of safe water standards for private well use. [Health]
4. You are not sure about the safety of genetically engineered foods, and would like to find more information and research on this topic. Name four potential types of safety problems that have been raised. [Health or Project]
5. You are interested in learning more about what measures the US government has taken since 2001 to prevent Mad-Cow Disease. Identify three such measures. [Health or Project]
6. Name/find three research programs/projects that investigate the treatment/causes of dwarfism. [Project]
7. You are planning a cycling expedition along the Silk Road in Central Asia. Find a website that is a good source information about health precautions should you take. [Travel]

8. You are planning to travel to the northeast territories of India and wonder if there are any problems/restrictions for tourists. Find a website that is a good source of information about such problems/restrictions. [Travel]

The various sites performed their own "grading" of the tasks and determination of relevant documents. An informal effort was made among groups to share answers and relevant documents.

A standard query was run by all searchers and the time between pressing the search button and the return of the results logged. This information was used to get an idea of the different response time conditions searchers may have encountered.

Experimental design

The experimental design followed the protocol developed for the TREC-9 Interactive Track. This design allowed the comparison of two systems or system variants. A minimum of 16 searchers were to be used. Each searcher was to perform all eight tasks (two of each of the four types), half on one system and half on another.

Each searcher was to issue the query "information retrieval" twice - once before beginning each set of the four searches on the two search engines in the design. The searcher ignored the results of the search. Experimenters collected elapsed time for these searches from the time the search button was pressed until results started to appear. This calibrating information was to be reported in each group's proceedings paper.

Evaluation

The searches were evaluated for effectiveness, efficiency, and user satisfaction in a manner similar to previous interactive tracks. Effectiveness included at least whether the task was completed successfully. Efficiency included at least the elapsed time used for each search. Instruments for the collection of minimal searcher background and satisfaction information were available and their use was encouraged.

Approaches

Here is a high-level description of the approaches taken by each group. For more detail, see the site report for each group.

CSIRO

In this year's Interactive Track, CSIRO continued to focus on answer organization issues, aiming to investigate whether the knowledge of "organizational structure" could be useful in organizing and delivering the retrieved documents. Particularly, for the collection of documents from the.gov domain, they used the level two domain name to categories the retrieved documents. For example, all documents from the nih.gov domain were put into the "National Institutes of Health" category. They compared this delivery method with the traditional ranked list. Their preliminary results indicated that there was no significant difference between the two delivery methods in the first 5 minutes, but the subjects performed significantly better with the category interface at the end of 15 minutes.

Oregon Health & Science University

The Oregon group initially planned to compare retrieval using alternative devices, e.g., a tablet device, but was not able to do so when the vendor was unable to deliver the devices in time. Instead, they chose to revisit the search for factors associated with successful searching. Their results identified some trends but the sample size was inadequate in size to achieve statistical significance.

Rutgers University

The Rutgers group investigated two major hypotheses: (a) that reducing the amount of interaction required of a searcher with the system leads to increased satisfaction with the search and increased performance, and (b) that increased query length leads to increased performance in the TREC 2002 Interactive Track task. Both of these hypotheses were the result of their investigations in the TREC 2001 Interactive Track.

They investigated the first hypothesis by implementing two different interfaces to the Panoptic search engine: one which displayed the default Panoptic result of 20 URLs and snippets at a time, requiring the searcher to follow links in order to view pages, and a second which displayed, in scrollable windows, four retrieved pages at a time. The latter was intended to reduce interaction effort in comparison to the former, by virtue of displaying retrieved pages directly. They investigated the second hypothesis by having two different versions of each of the two interfaces: one which labeled the query input box, "QUERY," and another which labeled it, "Information problem (the more you say, the better the search results are likely to be)." The demonstration of the first version of query elicitation used only words and phrases; the demonstration of the second version used full sentences and questions.

University of North Carolina Chapel Hill

The North Carolina group's research question was whether 3D visualization of search results was more effective than a text-based interface. Unlike most of the other interactive track groups, they used their our own locally developed software. The 3D visualization was a variation on an Information Space navigation system (as they have demonstrated at past TREC conferences); the text system was a fairly generic Google-style interface, not much different from the Panoptic system.

They hypothesized that people would be able to perform well with both the 3D and the text system, but would feel less confident with the 3D system. The 3D system had most of the same "controls" (text query input, display of results), which they hypothesized would be used similarly to the text system. They hypothesized that people would feel less confident with their 3D results. Finally, they hypothesized there would be no significant differences in results across the different search tasks, but anticipated slight ordering effects (increased performance over time with both systems).

University of Toronto

The primary objective of the Toronto group was to design a novel information exploration interface that combined multiple methods for accessing the content to accommodate the multiple perspectives that users bring to the task. Their work toward this goal was derived from the exploratory study done in conjunction with TREC 2001 Interactive Track. They had two goals for their TREC 2002 study:

1. Work toward a search workspace – a digital environment that contains a set of tools to aid users in exploring an information space
2. Develop a more cost-effective solution to information retrieval experimentation.

University of Glasgow

The Glasgow group assessed whether they could improve upon the limitations of the traditional elongated results list by an appropriate application of hierarchical clustering and summarization visualization techniques? Their current experimental system, provisionally named HuddleSearch, acted as an intermediate layer between the user and the provided Panoptic search engine. It used a newly developed clustering algorithm, which dynamically organizes the relevant documents into a traversable hierarchy of general to more specific clusters categories. They extended their TREC 2001 query-biased summarization tool to also allow the summarization of multiple documents, whereby a summary painted a caricature of the content of a cluster, rather than an individual document, thus allowing the user to provisionally judge a cluster's relevance prior to viewing its contents. The interaction between the user and the system was further developed by the aid of an information visualization tool.

From their initial analysis, they concluded that users do prefer the hierarchical clustering system to a list-based approach. Compared to the underlying Panoptic search engine, used as a baseline in their experimental design, the times taken to complete search tasks using their system clearly fell. In addition, the number of incomplete tasks was definitely reduced by the use of the experimental system.

Conclusions and Future Directions

While the track participants were pleased to be able to use Web data for experiments, a number of issues arose in this year's track. The main concern was the "open" .GOV collection, which made controlled experiments more challenging. Another problem was that the PDF files in the collection were proper consisted solely of text, which may be fine for batch experiments but is problematic for real users. Another general challenge for the track is the existence of high-quality commercial search engines such as Google, which make experimental systems frustrating for users when they do not perform as well as commercial products.

The track will undergo significant changes for TREC 2003. The track itself will become a subtrack of the Web track, and the current chair will be stepping down. Further details will be available on the TREC Web site.

References

1. Hersh WR, *Interactivity at the Text Retrieval Conference (TREC).* Information Processing and Management, 2001. 37: 365-366.
2. Wu M, Fuller M. and Wilkinson R. *Searcher performance in question answering. Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* 2001. New Orleans, LA: ACM Press. 375-381.
3. Belkin NJ, et al., *Iterative exploration, design and evaluation of support for query reformulation in interactive inforniation retrieval.* Information Processing and Management, 2000. 37: 403-434.
4. Hersh W, et al. *Do batch and user evaluations give the same results? Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* 2000. Athens, Greece: ACM Press. 17-24.
5. Hersh W and Over P, *SIGIR Workshop on Interactive Retrieval at TREC and Beyond.* 2000, SIGIR Forum, http://www.acm.org/sigir/forum/S2000/Interactive_report.pdf.

Overview of the TREC 2002 Novelty Track

Donna Harman
National Institute of Standards and Technology
Gaithersburg, MD 20899

Abstract

The novelty track was a new track in TREC-11. The basic task was as follows: given a TREC topic and an ordered list of relevant documents (ordered by relevance ranking), find the relevant and "novel" sentences that should be returned to the user from this set. There were 13 groups that participated in this new task.

1 Introduction

The novelty track is a new track in TREC this year and therefore both the results and the evaluation should be viewed as a pilot study. The direct motivation for this track (and the track name) came from Prof. Jaime Carbonnell's talk to the Automatic Summarization Workshop at NAACL in May 2001. In this talk he mentioned that there were other ways to optimize search results than just using relevance ranking alone. One could rank on timeliness of the article, validity of the source of the article, and the comprehensibility and NOVELTY of the information in the article to the user. Whereas the first three of these optimization characteristics are either trivial (timestamps) or very difficult (validity of source and comprehensibility to user), the novelty issue could be operationalized for evaluation by assuming that a user knows nothing at the time of the initial relevant document and all learning happens in the order of document retrieval.

This was the basic design of the novelty track. Given a TREC topic and an ordered list of relevant documents (ordered presumably by relevance ranking alone), find the "novel" information that should be returned to the user from this set. However, unlike earlier work with novelty and redundancy in documents [2], the decision was made to tackle this task at a sentence level. There are several reasons for this decision. First, it was hoped that by reducing the granularity of text unit, it would be easier to identify novel or redundant information. But equally important, the use of sentences combined the novelty task with earlier suggestions for evaluation of passage retrieval. Because the size of a passage is not easily defined (paragraphs are not always available), it was decided to define the "passages" at a sentence level, where the demarcation is clear.

A possible application scenario here would be to envision a smart "next" button that would allow a user to walk down a ranked list of documents by highlighting the next relevant and NOVEL sentence. The user could then view that sentence and if interested, also read the surrounding sentences. Alternatively this task could be viewed as finding key sentences that could be useful as "hot spots" for collecting information to summarize an answer of length X to the information request.

Note that this track is another effort to get beyond the ranked list output for information retrieval. The TREC question-answering track is one approach, but only for direct questions and only for short, fact-based questions. The novelty track explores an alternative approach by returning only relevant AND novel sentences rather than whole documents containing extraneous or duplicate information.

2 Input Data and Task Definition

The basic input data for the novelty track was a set of 50 topics taken from TRECs 6, 7, and 8 (topics 300-450). These 50 topics were selected from the full set of 150 by picking those topics that had between 10 and 70 relevant documents, plus eliminating a few that had large numbers of Federal Register documents, which tend to be very long. This choice was based on having enough relevant documents to work with but not too many for humans to process in creating the truth data. (After NIST staff tried the task, the maximum number of documents was determined to be no more than 25!)

To select the documents once the topics had been picked, NIST modeled the application by selecting actual results from an effective manual run from the appropriate TREC. If there were 25 or fewer relevant documents for the topic, then all the relevant documents were used. If there were more than 25 documents, the top 25 ranked (and relevant) documents from that run were selected for the task. If the run did not find 25 relevant documents, then all the relevant documents it did find were included, along with a random sample of the missing relevant documents up to a maximum of 25 documents. In all cases documents from the Congressional Record (also very long) were NOT selected.

Once the documents were selected, they were ranked at NIST using the ordering produced by same run used in selecting the documents (where possible). Each document was also automatically split into sentences at NIST and sentences were assigned identifiers.

Participants were provided with the topics, the set of sentence-segmented documents, and the order of retrieval for those documents, and were required to process the documents in this order. Each run was to output two ordered sets of sentence identifiers for each of the 50 topics. The first set of sentences was the set of sentences the system determined to contain relevant information. The second set of sentences was the subset of those relevant sentences that the system determined to contain new information, that is, relevant information that was not contained in earlier sentences.

The task was to be done completely automatically. Any fields in the topic could be used, along with any other resources. It was assumed that the set of relevant documents was available as an ordered set, i.e. the entire set could be used in deciding the sentence sets. However the topics had to be processed independently. Both these restrictions reflect the reality of the application.

The content of the topics was a modified version of the original TREC topic statement for each topic, including the original topic fields plus an additional description field (tagged DESC2). This field was what the assessor used as the information need to build the relevant (and novel) sets. Often this was the same as the original description, but many times it was not. The assessors were not supposed to use the narrative field and often the revised description includes some piece of the narrative that they did use.

The test data was released on June 21, 2002, with results due at NIST on September 3.

3 Evaluation

3.1 Creation of truth data

Judgment data was created by having NIST assessors manually perform the task. First they created a file of the relevant sentences and then reduced that file to those that were novel given the document order. Both files were saved. Each topic was independently judged by two different assessors so that the effects of different human opinions could be assessed. Figure 1 presents the exact instructions given to the assessors.

3.2 Analysis of truth data

Since the novelty task was basically requiring systems to automatically select the same sentences that were selected manually by the assessors, it is important to analyze the characteristics of the manually-created truth data in order to better understand the system results. Five particular aspects were examined.

1. what percentage of the sentences in the relevant documents were marked relevant and how does this vary across topics and across assessors?
2. what percentage of the relevant sentences were marked novel and how does this vary across topics and across assessors?
3. how well were the assessors able to pick only "key" sentences as opposed to gathering many consecutive sentences for each relevant piece of information?
4. how different are the results of the two assessors for each topic?
5. where do these differences occur?

Table 1 shows the number of relevant sentences and novel sentences selected for each topic by each of the two assessors that worked on that topic. The column marked "minimum" precedes the results for the assessor who selected the fewest number of relevant sentences. The column marked "rel" is the number of sentences selected as relevant; the next column is the percent of sentences from the total set of relevant documents for that topic that were selected as relevant. The last two columns for each assessor show the number of sentences marked novel and the percentage of the relevant sentences that were marked novel.

Whereas there are large differences in the number of relevant sentences for each topic, there is only a weak topic effect on the percentage of total sentences

Figure 1: Instructions to Assessors

You are trying to create a list of sentences that are:

1. relevant to the question or request made in the description section of the topic,
2. their relevance is independent of any surrounding sentences,
3. they provide new information that has not been found in any previously picked sentences.

Instructions to assessors

1. order printed documents according to the ranked list
2. using the description part of the topic only, go thru each printed document and mark in yellow all sentences that directly provide information requested by the description. Do not mark sentences that are introductory or explanatory in nature. In particular, if there is a set of sentences that provide a single piece of information, only select the sentence that provides the most detail in that set. If two adjacent sentences are needed to provide a single piece of information because of an unusual sentence construction or error in the sentence segmentor, mark both.
3. go to the computer and pull up the online version of your documents. Go through each document, selecting the sentences that you have previously marked (you can change your mind). Save this edited version as "relevant"
4. now go thru the online version looking for duplicate information. Order is important here; if a piece of information has already been picked, then repeats of that same information should be deleted. Instances that give further details of that information should be retained, but instances that summarize details seen earlier should be deleted. Save this second edited version as "new".

per topic selected as relevant. Looking at the results from the "minimum" assessor, the median percentage of sentences marked relevant is about 2%, with the range from 0.2% to 6%. Note that this number is confounded with the assessor effect, as there is a very strong relationship between number of relevant sentences selected and the assessor (see Table 2 for more on this).

It is somewhat surprising that so few of the sentences were selected as relevant. Again using the minimum assessor, 16 of the 50 topics had fewer than 1% of the sentences selected as relevant, whereas only 3 of the 50 topics had more than 5% of the sentences marked relevant.

Equally surprising is the high percentage of relevant sentences that were marked as novel. The median percentage of relevant sentences marked as novel was 93%, with the range from 50% to 100%. In fact, 23 of the 50 topics had ALL relevant sentences marked as novel (by the "minimum" assessor). The fact that the assessors judged most sentences as being novel was a major disappointment of the track. One postulated cause is that the documents being judged were mostly from different sources and different time periods and therefore there was truly little duplicate information. However, it is also possible that judgments for novelty at the sentence level (for long sentences) are likely to always find new information. Next year's novelty track will use multiple sources reporting on the news within the same time period in the hope that a lower percentage of relevant sentences will be marked novel.

Table 2 demonstrates the assessor effect more clearly. The table is ordered by the percentage of the sentences that were marked relevant. The column marked "min" shows the number of topics for which the given assessor had the fewest number of relevant sentences. The next two columns present the total number of relevant sentences found by that assessor and the percentage of sentences in the relevant documents that were marked relevant. The columns marked novel and %rel give the total number of sentences marked novel and the percentage this was of the number marked relevant. The last two columns

Table 1: Analysis of relevant and novel sentences by topic

Topic	minimum	rel	%total	novel	%rel	maximum	rel	%total	novel	%rel
305	A	15	2.01	15	100	B	51	6.84	49	96.08
310	C	0	0	0	0	D	40	7.08	36	90
312	C	5	0.87	5	100	A	18	3.12	12	66.67
314	E	25	2.35	25	100	F	54	5.08	52	96.3
315	C	18	3.08	11	61.11	F	54	9.25	54	100
316	B	22	0.99	18	81.82	G	49	2.2	46	93.88
317	C	23	4.19	23	100	B	33	6.01	22	66.67
322	G	34	4.57	34	100	A	96	12.9	84	87.5
323	G	65	4.57	60	92.31	A	88	6.15	86	97.72
325	G	21	1.25	21	100	F	45	0	lost	0
326	C	10	1	8	80	G	74	7.39	61	82.43
330	E	29	3.49	27	93.1	B	38	4.57	29	76.32
339	C	12	1.6	11	91.67	E	26	3.46	25	96.15
342	E	17	2.17	17	100	G	81	10.32	71	87.65
345	C	47	5.19	47	100	B	65	7.18	52	80
351	C	6	0.75	5	83.33	E	31	3.87	28	90.32
355	F	103	3.94	78	75.73	D	223	8.53	200	89.69
356	D	10	1.2	9	90	B	19	2.29	18	94.74
358	C	40	4.8	37	92.5	E	55	6.6	46	83.64
362	B	47	5.15	46	97.87	A	71	7.79	65	91.55
363	C	11	2.08	10	90.91	A	45	8.52	35	77.78
364	C	42	3.5	42	100	E	45	3.75	45	100
365	G	34	2.8	34	100	E	41	3.38	41	100
368	G	71	4.63	66	92.96	D	121	7.89	94	77.69
369	B	13	1.79	12	92.31	F	30	4.13	25	83.33
377	G	3	0.19	3	100	E	25	1.56	22	88
381	G	19	1.38	19	100	B	38	2.76	29	76.32
382	C	41	1.83	24	58.53	E	114	5.07	110	96.49
384	G	23	1.41	23	100	A	124	7.57	111	89.52
386	A	43	4.3	41	95.35	G	43	4.3	43	100
388	E	56	4.57	56	100	F	123	10.04	96	78.05
394	G	21	2.45	21	100	B	28	3.27	25	89.29
397	C	29	6.18	28	96.5	E	32	6.82	30	93.75
405	B	40	3.75	37	92.5	F	75	7.02	70	93.33
406	G	10	1.79	10	100	C	12	2.14	10	83.33
407	B	32	2.46	29	90.62	A	1301	100	58	4.46
409	B	17	3.26	12	70.59	G	27	5.17	25	92.59
410	E	17	1.92	15	88.24	F	83	9.39	53	63.86
411	C	21	1.86	19	90.48	A	60	5.31	53	88.33
414	E	29	3.62	25	86.21	A	31	3.87	18	58.06
416	E	36	1.96	30	83.33	F	46	2.51	38	82.61
419	A	50	3.32	36	72	D	50	3.32	41	82
420	C	18	1.4	18	100	G	27	2.1	23	85.19
427	E	14	0.31	11	78.57	F	58	1.29	48	82.76
432	E	9	0.96	8	88.89	B	33	3.53	27	81.82
433	C	11	1.72	7	63.64	F	23	3.59	23	100
440	E	19	1.35	19	100	G	107	7.62	98	91.59
445	F	10	1.07	5	50	E	13	1.4	7	53.85
448	C	20	2.25	20	100	D	41	4.62	38	92.68
449	E	57	3.65	57	100	F	81	5.19	71	87.65

Table 2: Analysis of relevant and novel sentences by assessor

Assessor	min	rel	%total	novel	%rel	consecutive	%rel
C	17	348	0.60	343	98.56	97	0.2787
B	6	476	0.82	405	85.08	129	0.2710
D	1	485	0.84	418	86.19	267	0.5505
E	11	690	1.19	644	93.33	193	0.2797
G	10	709	1.23	658	92.81	261	0.3681
F	2	740	1.28	658	88.92	394	0.5324
A	3	1940	3.36	616	31.75	1498	0.7722

present the total number of relevant sentences that were consecutive, and this percentage.

As can be seen from Table 2, there is a major effect from the assessors. For example, assessor "C" was the "minimum" assessor for 17 of the 50 topics; assessor "D" was the minimum assessor for only 1 of the topics. Part of this effect is the normal human variation in relevance judgments as some judges are stricter than others or interpret the question differently. However a second part of the variation comes from different interpretations of the instructions for the task. Note that there is a distinct trend towards more consecutive sentences as there are more relevant sentences selected.

Summarizing the answers to the first three questions:

1. A very low percentage (median 2%) of the sentences in the relevant documents were marked relevant. There was small (0.2% to 6%) but random effect across topics, but definite trends across the assessors in terms of the percentage of sentences (and documents) marked relevant. This is consistent with past findings in the judgment of relevant documents.

2. A very high percentage (median 93%) of the relevant sentences were marked novel. Here there was more variation across topics and less across assessors.

3. In general the assessors were not able to pick only "key" sentences as opposed to gathering consecutive sentences for each relevant piece of information. The percentage of relevant sentences that were consecutive (before or after another relevant sentence) varied from a low of 30% to a high of 77% across assessors.

The final two questions, how different is the output of the two assessors for each topic and where do these differences occur, are explored by Table 3. This table analyzes the differences between the two assessors' relevant sentence judgments. The first two columns give the number of relevant sentences selected by the minimum and maximum assessor for each topic. The next two columns give the sentence coverage and overlap between the two assessors. The overlap is the intersection between their sentences divided by the union of their sentences, i.e. the percentage of the matching sentences divided by sentences that either assessor had selected. The coverage is the intersection of the sentence sets divided by the smaller of those sets, i.e. the percentage of the minimum assessor's sentences that were also chosen by the maximum assessor. The column marked d_cover is the coverage between the documents containing sentences marked relevant, while d_over is the corresponding overlap. The final two columns give the percentage of consecutive sentences selected by the minimum and maximum assessor.

The table is sorted in descending coverage order. Since the minimum assessor has been designated as the "official" assessor in terms of scoring, the coverage metrics are particularly interesting in this evaluation. The average coverage is 0.579, with a median of 0.61. This means that the "second" assessor, in this case the one that picked the larger number of relevant sentences, picked about 60% of the official assessors sentences, plus often many more sentences.

This seemingly low figure is remarkedly similar to that found for relevance judgments for full documents[1], where relevance assessors were shown to agree about 60% of the time that a given document was relevant. But for the novelty task, there are different factors feeding into this disagreement. There is the interpretation of the question and the relative strictness of the judges, factors that are seen in the judgments of full documents. But additionally there is also the issue of how large a section of a given document to select, a factor measured by the number of consecutive sentences.

It was hoped that there would be some clear correlation between some of the measures in Table 3. For example, a low overlap would be correlated with a high number of consecutive sentences. However this is not the case; plots of the coverage or overlap versus the other factors resemble random scatterplots. Illustrations of this can be seen by looking at some of the topics. Topic 427, with a perfect coverage score of 1.00, has an overlap of 0.24. This is due to a large difference in the number of consecutive sentences selected, but ALSO to selection of different sentences as measured by the low document overlap (0.53). Topic 314 has similar number of consecutive sentences selected by each assessor, but the assessors picked different sentences resulting in a low coverage (0.44) and an even lower overlap (0.16).

The complex set of factors governing the differences between assessors makes it unlikely that these differences will lessen. In particular, the assessor instructions to avoid the use of consecutive sentences did not work, and it is unlikely that agreement would have improved even if there had been fewer consecutive sentences. In the next running of the novelty track this instruction will be removed.

What remains to be done given this low level of agreement is to understand how the system comparisons are effected by all this noise. Voorhees [1] showed that there was no effect as long as enough topics were used for averaging. Part of the analysis next year will be to check if this also holds true for sentence relevance in the novelty track.

3.3 Scoring

The sentences selected manually by the NIST assessors were considered the truth data. Obviously we could have chosen one set of assessors as the official one (similar to TREC ad hoc), and use the second set only for human agreement measurements. However the decision was made to use the truth data from the assessor that marked the smallest number of relevant sentences (on a per topic basis) as the main score. The reason for this is that the biggest disagreement between assessors had to do with how many sequential sentences they took as relevant. Often they included sentences "for context", even though the instructions tried to discourage this. By taking the minimum of two assessors, it was hoped to avoid many of the disagreements. This definition also matches better with the stated goals of the track. Note that one assessor disagreed with the original assessor's relevance judgments for topic 310 and could find no relevant sentences in any of the documents. We eliminated that topic from the final test set, so scores were computed over the remaining 49 topics.

Participants were told that the scoring would be based on the smaller set before runs were submitted, but, of course, they did not have access to the assessor sentence sets.

The track guidelines specified sentence set recall and precision as the evaluation measures for the track. Let M be the number of matched sentences, i.e., the number of sentences selected by both the assessor and the system, A be the number of sentences selected by the assessor, and S be the number of sentences selected by the system. Then sentence set recall is M/A and precision is M/S.

As previous filtering tracks have demonstrated, set-based recall and precision do not average well, especially when the assessor set sizes vary widely across topics. Consider the following example as an illustration of the problems. One topic has hundreds of relevant sentences and the system retrieves 1 relevant sentence. The second topic has 1 relevant sentence and the system retrieves hundreds of sentences. The average for both recall and precision over these two topics is approximately .5 (the scores on the first topic are 1.0 for precision and essentially 0.0 for recall, and the scores for the second topic are the reverse), even though the system did precisely the wrong thing. While most real submissions won't exhibit this extreme behavior, the fact remains that recall and precision averaged over a set of topics is not a good diagnostic indicator of system performance. There is also the problem of how to define precision when the system returns no sentences ($S = 0$). Not counting that question in the evaluation for that run means different systems are evaluated over different numbers of topics, while defining precision to be either 1 or 0 is extreme. (The average scores given in Appendix A defined precision to be 0 when $S = 0$ since that seems the least evil choice.)

To avoid these problems, the primary measure reported for novelty track runs is the F measure, defined as

$$\mathrm{F} = \frac{2 \times \mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}}$$

The average of the F measure is meaningful even when the judgment sets sizes vary widely. For example, the F measure in the scenario above is essentially 0, an intuitively appropriate score for such behavior. Using the F measure also deals with the problem of what to do when the system returns no sentences since recall is 0 and the F measure is legitimately 0 regardless of what precision is defined to be.

Table 4: Organizations participating in the TREC 2002 Novelty Track

Carnegie Mellon University
Columbia University
Fudan University
IRIT/SIG
National Taiwan University
NTT Communication Science Labs
Queens College, CUNY
Streamsage
Tsinghua University
University of Amsterdam/ILLC
University of Iowa
University of Massachusetts at Amherst
University of Michigan

4 Participants and Descriptions of Approaches

Table 4 lists the 13 groups that participated in the TREC 2002 novelty track. The rest of this section contains short summaries submitted by most of the groups about their approaches to the novelty task.

4.1 Carnegie Mellon University

To find relevant sentences we used a simple baseline of cosine similarity with tf.idf weighting and pseudo-relevance feedback, treating sentences as very short documents. We tried different classifiers using lexical and semantic features derived from a simple parse as well as proximity to sentences with very high tf.idf scores. To model redundant sentences we used a very simple translation model. The translation probabilities for word or short phrase pairs were based on the skew divergence between word distributions derived from a mixture model of unigrams extracted from WordNet relations. The parse tree for each sentence was transformed into a graph of modifier relations. The overall redundancy measure between sentences was then calculated using a basic greedy graph-matching algorithm.

4.2 Columbia University

Our principal interest in the Novelty Track was to experiment with ideas we are developing in the detection of new information, but we found that the relevance part of the task here absorbed most of the time that we had alloted and most of our attention. We decided that it would be more interesting to adapt our new information tools to the relevance part than to try to use established IR strategies. Our approach on relevance question was to expand the information in all fields of the given topics with 1) semantic equivalents of the content words in the topic, and 2) related words determined by co-occurrence statistics from a background corpus. Sentences were selected largely by the number of words that match the expanded queries. In addition, we reclustered the set of relevant documents and in some cases eliminated many documents from inclusion. We had little time left for the novelty part and relied solely on the overlap of the topic words and their semantic equivalents.

4.3 IRIT/SIG

IRIT developed a new strategy in order to detect the relevant sentence that we did not try in a more general context of document retrieval but did try previously in document categorization. In our approach a sentence is considered as relevant if it matches the topic with a certain level of coverage. This level of coverage depends on the category of the words. Three types of words have been defined: highly relevant, lowly relevant and no relevant. With regard to the novelty part, a sentence is considered as novel with regard to a topic when its level of coverage with the previously processed sentences and with the best-retrieved sentences does not exceed a certain threshold.

4.4 National Taiwan University

In the novelty task, the amount of information that can be used in a sentence is the major challenging issue. Some sort of information expansion method was introduced to tackle this problem. Our approach to relevance identification was to expand the information of a sentence with the context of this sentence using a sliding window method. The similarity was measured by the number of words of a topic description that match the sentences within a window. Besides, WordNet was employed to relax word match operation to inexact match. In the novelty detection part, we first applied a coherent text segmentation algorithm to partition the sentences extracted from the relevance identification part into several coherent passages denoting sub-topics. Then we compute the similarity of each sentence with each passage. A sentence was in terms of a sentence-passage similarity vector. Two sentences are regarded as similar if they are related to the same sub-topics . In this way, the redundant sentences were filtered out.

4.5 NTT Communication Science Labs

Our approach is based on query-biased multi-document summarization methods. “Relevant” sentences are selected from each document using Support Vector Machines(SVMs), which are trained on a query-sentence data set. The data set consists of query-sentence pairs whose relevance are judged by us and the queries are chosen from the TREC topics that are not used in the novelty track. “New” sentences are chosen from the relevant sentences based on Maximum Marginal Relevance (MMR) measure.

4.6 Queens College, CUNY

For this experiment, we employ all sections of a topic to form long queries for retrieval because the “documents” are actually sentences. The queries average to 938/49 unique terms. Since the sentences come from relevant documents of TREC-8, we use the TREC-8 dictionary to provide better statistics for processing and retrieval. However, the high Zipf threshold has been reset to 400,000 to include more high frequency terms.

Only initial retrieval without pseudo-relevance feedback was performed. Based on experimentation with the four training topics, we decide to test two RSV threshold (tr) values to help decide on relevance of retrieved sentences: submission pircs2N01/2 use tr=1.25, and pircs2N03/4 use tr=1.5.

This set of relevant sentences is sorted according to DocID. For each sentence, every one of its unstemmed words is expanded with synonyms by consulting with WordNet. The resultant set of words is sorted and duplicates removed. A double loop passes down the sentence list, and a novelty coefficient based on the Dice formula is evaluated for each pair of sentences S_i and S_j. The novelty coefficient is defined as the intersection of S_i and S_j divided by the union of S_i and S_j.

If the novelty coefficient is less than a threshold tv, S_j is considered novel with respect to S_i, otherwise S_j is removed. pircs2N01 and pircs2N03 employ a threshold tv=0.3, and pircs2N02, pircs2N04 use tv=0.5. In addition, a fifth submitted run pircs2N05 does not use synonyms, just raw words, and acts as control. Its thresholds are tr=1.5, tv=0.3.

4.7 Streamsage

I concentrated on query expansion, using our New York Times news story corpus, and the description field. Starting with the nouns in the topic title, I searched our corpus for multi-word units containing them (subject to grammatical and frequency constraints), then added other nouns in the MWU as search terms. I also included nouns from desc in the same noun phrase as a title noun. Sentences which contained a search term were returned as relevant.

4.8 Tsinghua University

In this year’s novelty track, we performed two-step research to find relevant sentence, and then to eliminate repetitive information. On finding relevant information, our work focused on four parts:

1. Extracting key information in topics. We classified words in the topic into three classes by statistical learning and rule-based learning: useful keywords, general describing words and negative words.

2. Finding efficient query expansion (QE) techniques. Besides thesaurus based QE, we proposed and studied a new statistical expansion approach, which expands terms that co-occurred in a fixed window size with title words in the relevant document set, called local co-occurrence expansion. The results are extremely good.

3. document/sentence term expansion (DE). Sometimes, the query mentions a general topic while some relevant documents describe detailed information. In this case, QE may not help because you do not know to what extent the terms should be expanded. We proposed term expansion in documents (referred as DE) to solve the problem.

4. topic classification. QE and DE are oriented from two aspects of retrieval problem and may work well for different types of topics. Therefore we classified the topics into two classes according to similarities between topic fields to perform QE or DE respectively, which leads to better performance than either approach. On eliminating repetitive information, rather than concept of similarity, we used the concept of unsymmetrical sentence overlapping. It represents the extent of the information taken by one sentence overlapped by another one. Our experimental results show it is better than the symmetrical measure

of similarity. Two different elimination strategies are studied. One is sentence to sentence comparison, the other one is sentence to pool overlapping technique. In our experiments, the performances of two strategies are almost equal.

4.9 University of Amsterdam/ILLC

For identifying *relevant* sentences, we used a fairly minimal approach. For a given topic, the sentences in the relevant documents for that topic were viewed as documents themselves, and we ran the topic against this sentences-as-documents collection using a retrieval engine based on a standard vector space model, with the tfv.nfx weighting scheme. Three different runs were submitted: one where all documents were stemmed, a second where they were lemmatized, and in the third run the results of the other two runs were simply merged.

For identifying the list of *new* sentences, we scanned the list of relevant sentences and filtered out sentences that were entailed by the sentences kept so far. Our notion of entailment between two text segments s_1 and s_2 is based on comparing the sum of the inverted document frequencies of the terms that occur in both s_1 and s_2 to the inverted document frequencies of the terms occurring in s_2. If it is beyond a certain threshold, this entailment score prevents a sentence s_2 from being added to the set of new sentences s_1 that we have built up so far.

4.10 University of Massachusetts at Amherst

Our approach was to apply standard techniques that have proven successful for document retrieval and filtering to see if they also work well at the sentence level. We began by building a larger training corpus from 48 of the TREC ad-hoc retrieval track topics, supplementing the handful of training topics that NIST provided. Our methods for building this corpus followed the general specifications for building the test collection. We used the same instructions provided to the NIST assessors, though we used undergraduates rather than retired analysts to do the assessments.

The task was then treated as two separate problems: (1) identify the relevant sentences then (2) filter out the redundant sentences. In identifying relevant sentences, we experimented with several retrieval models, including language modeling and the vector space model with tf-idf weighting. We found that the tf-idf approach worked best on our training data, so both of our systems (CIIR02tfkl and CIIR02tfnew) use that method to identify relevant sentences.

For novelty filtering, we built two different systems. One (CIIR02tfkl) uses the Kullback-Leibler divergence between a sentence and all previously seen sentences to assign novelty scores. The other (CIIR02tfnew) employs a set-difference approach by counting the number of previously unseen words in each relevant sentence. On both the training and the test data, the set difference system outperformed the language modeling system when applied to our own relevance results. However, for both collections, the language modeling approach performed better when applied to the known relevant sentences–i.e., if we "cheat" and use the truth data as a preliminary step.

4.11 University of Michigan

The Michigan novelty detection systems for this year's evaluation were based on the MEAD multi-document, extractive summarizer. Using the sample data, we first concentrated on modifying MEAD to better detect sentences that are relevant to a user's query. In particular, our modifications tried to capture our observation that the humans who judged the sample clusters tended to choose groups of relevant sentences, rather than individual sentences from different places in a source document. To do this, we implemented a new sentence reranker within MEAD, which favored groups of sentences with relatively high concentrations of key words relevant to the overall cluster of documents. We also developed some new sentence features within MEAD, which measured how related a given sentence is to a user's query. Specifically, we used a query-title-word-overlap feature, which quanified the extent to which a given sentence contained words that were present in the title of the user query.

Our aim main in participating in the novelty track was to set a simple base line for future, linguistically motivated experiments on the task.

5 Results

Thirteen groups submitted 43 runs to the novelty track. For all runs, the F score for the relevant sentence sets was greater than the score for the new sentence sets. This suggests that finding the relevant parts of a document is somewhat easier than finding the nonredundant parts.

Since the novelty track was a completely new task, groups had no training data and little idea of what to expect. One group (the University of Massachusetts

Table 5: Average F scores for baseline and system results for the Novelty track.

	Relevant	New
Second human judges	0.371	0.353
Random sentences	0.040	0.036
thunv1	0.235	0.217
thunv2	0.235	0.216
thunv3	0.235	0.216
CIIR02tfnew	0.211	0.209
thunv4	0.225	0.206
CIIR02tfkl	0.211	0.196
pircs2N02	0.209	0.193
pircs2N01	0.209	0.188
pircs2N04	0.197	0.184
ss1	0.186	0.183

at Amherst) constructed extensive training data following the assessor instructions and used this to guide their research (run tags CIIR). Another high scoring group, City University of New York (run tags pircs), used traditional information retrieval methods, treating the sentences as documents. Tsinghua University (run tags thunv) used a completely new method devised especially for this task.

One of the requirements for a new track is to do sanity-checking of the evaluation itself. To this end, NIST computed the average F score for the second human assessor sentence sets and for sets of sentences randomly selected from the target documents. The results are shown in Table 5, which also includes the scores for some of the best runs for comparison. The scores for the systems fall in between the human and random performance, support for a claim that the evaluation is credible.

The track will be run again in 2003, with topics specifically constructed for the task. The data will consist of several news sources from the same time period in hopes that there will be more duplicate information.

References

[1] Ellen M. Voorhees. Variations in relevance judgments and the measurement of retrieval effectiveness. *Information Processing and Management*, 36(5):697–716, 2000.

[2] Yi Zhang, Jamie Callan, and Thomas Minka. Novelty and redundancy detection in adaptive filtering. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 81–88, 2002.

Table 3: Analysis of sentence coverage and overlap, document coverage and overlap and percent of consecutive sentences by topic

Topic	min.rel	max.rel	coverage	overlap	d_coverage	d_overlap	%min consecutive	%max consecutive
427	14	58	1.00	0.24	1.00	0.53	7.14	41.38
407	32	1301	1.00	0.02	1.00	0.52	34.38	98.08
397	28	32	0.89	0.71	1.00	1.00	7.14	15.62
315	18	54	0.89	0.29	1.00	0.40	11.11	27.78
305	15	51	0.87	0.25	0.83	0.62	40	58.82
365	34	41	0.82	0.60	0.81	0.65	29.41	24.39
414	29	31	0.79	0.62	0.95	0.83	17.24	9.68
440	19	107	0.79	0.14	0.91	0.42	10.53	48.6
364	42	45	0.74	0.55	0.90	0.82	30.95	42.22
433	11	23	0.73	0.31	0.83	0.45	9.09	43.48
355	103	223	0.73	0.30	0.94	0.62	54.37	69.51
322	34	96	0.71	0.23	0.75	0.52	23.53	43.75
419	50	50	0.70	0.54	1.00	0.88	30	22
358	40	55	0.68	0.40	1.00	0.72	22.5	16.36
368	71	121	0.68	0.33	0.96	0.88	39.44	61.16
432	9	33	0.67	0:17	0.88	0.39	11.11	27.27
382	24	114	0.67	0.13	1.00	0.41	20.83	37.72
377	3	25	0.67	0.08	0.67	0.29	0	52
362	47	71	0.66	0.36	1.00	1.00	17.02	15.49
405	40	75	0.65	0.29	0.94	0.75	22.5	50.67
448	20	41	0.65	0.27	0.80	0.50	15	29.27
388	56	123	0.64	0.25	0.78	0.64	39.29	73.17
363	11	45	0.64	0.14	0.70	0.44	9.09	22.22
330	29	38	0.62	0.37	1.00	0.87	17.24	21.05
384	23	124	0.61	0.11	0.89	0.36	21.74	50.81
312	5	18	0.60	0.15	1.00	0.10	40	0
326	10	74	0.60	0.08	1.00	0.20	30	45.95
410	17	83	0.59	0.11	0.93	0.62	5.88	45.78
339	12	26	0.58	0.23	1.00	0.67	8.33	42.31
323	65	86	0.55	0.31	0.94	0.73	46.15	50
369	13	30	0.54	0.19	1.00	0.80	15.38	43.33
342	17	81	0.53	0.10	0.92	0.67	11.76	60.49
406	10	12	0.50	0.29	0.83	0.56	0	33.33
356	10	19	0.50	0.21	0.83	0.42	10	15.79
351	6	31	0.50	0.09	0.83	0.33	0	25.81
394	21	28	0.48	0.26	0.83	0.62	28.57	25
317	23	33	0.48	0.24	1.00	0.70	47.83	36.36
409	17	27	0.47	0.22	0.85	0.52	11.76	0
345	47	65	0.45	0.23	1.00	0.64	38.3	21.54
314	25	54	0.44	0.16	1.00	0.32	44	42.59
386	43	43	0.40	0.25	1.00	0.88	32.56	20.93
416	36	46	0.39	0.21	0.83	0.48	11.11	39.13
411	21	60	0.38	0.11	0.86	0.30	33.33	25
449	57	81	0.37	0.18	0.86	0.40	35.09	80.25
445	10	13	0.30	0.15	1.00	0.40	40	7.69
381	19	38	0.26	0.10	0.80	0.63	0	21.05
325	21	45	0.00	0.00	1.00	0.47	9.52	0
420	18	27	0.00	0.00	1.00	0.05	83.33	18.52
310	0	40	0.00	0.00	0.00	0.00	35	0

Overview of the TREC 2002 Question Answering Track

Ellen M. Voorhees
National Institute of Standards and Technology
Gaithersburg, MD 20899

Abstract

The TREC question answering track is an effort to bring the benefits of large-scale evaluation to bear on the question answering problem. The track contained two tasks in TREC 2002, the main task and the list task. Both tasks required that the answer strings returned by the systems consist of nothing more or less than an answer in contrast to the text snippets containing an answer allowed in previous years. A new evaluation measure in the main task, the confidence-weighted score, tested a system's ability to recognize when it has found a correct answer.

The goal of the question answering (QA) track is to foster research on systems that retrieve answers rather than documents in response to a question, with particular emphasis on systems that can function in unrestricted domains. Now in its fourth year, the tasks in the track have evolved over the years to increase the realism of the task and to focus research on particular aspects of the problem deemed important to improving the state-of-the-art. All of the tasks have involved finding answers to closed-class questions within a large corpus of news text.

This paper provides an overview of the TREC 2002 QA track. This year's track contained two tasks, the main task and the list task. Both tasks were also run in TREC 2001, but systems were required to return exact answers this year. That is, the text string returned by the system in response to a question was required to consist of a complete answer and nothing else, in contrast to earlier years where systems could return text strings that simply contained an answer. To make the paper self-contained, the first section recaps the tasks and evaluation procedures used in the first three tracks. The following sections then describe this year's tasks.

1 Evolution of the TREC QA Track

The task in the first two QA tracks (TRECs 8 and 9) was the same. For each question in the question set, systems retrieved a ranked list of up to five text snippets that contained an answer to the question plus a document that supported the answer. The collection of documents from which the support was drawn was a large set of newswire and newspaper articles. The questions were restricted to factoid questions such as *In what year did Joe DiMaggio compile his 56-game hitting streak?* and *Name a film in which Jude Law acted.* Each question was guaranteed to have at least one document in the collection that explicitly answered it. The maximum length of the text snippets was either 50 or 250 bytes, depending on the run type.

Human assessors read each string and decided whether the string actually did contain an answer to the question in the context provided by the document. Given a set of judgments for the strings, the score computed for a submission was the mean reciprocal rank. An individual question received a score equal to the reciprocal of the rank at which the first correct response was returned, or zero if none of the five responses contained a correct answer. The score for a submission was then the mean of the individual questions' reciprocal ranks.

The TREC-8 track both defined how answer strings were judged, and established that different assessors have different ideas as to what constitutes a correct answer even for the limited type of questions used in the track. A [*document-id*, *answer-string*] pair was judged correct if, in the opinion of the assessor, the answer-string contained an answer to the question, the answer-string was responsive to the question, and the document supported the answer. If the answer-string was responsive and contained a correct answer, but the document did not support that answer, the pair was judged "Not supported". Otherwise, the pair was judged incorrect. Requiring that the answer string be responsive to the question addressed a variety of issues. Answer strings that contained multiple entities of the same semantic category as the correct answer but did not indicate which of those entities was the actual answer (e.g., a list of names in response to a who question) were judged as incorrect. Certain punctuation and units were also required. Thus "5 5 billion" was not an acceptable substitute for "5.5 billion", nor was "500" acceptable when the correct answer was "$500". Finally, unless the question specifically stated otherwise, correct responses for questions about a famous entity had to refer to the famous entity and not to imitations, copies, etc. For example, two TREC-8 questions asked for the height of the Matterhorn (i.e., the Alp) and the replica of the Matterhorn at Disneyland. Correct responses for one of these questions were incorrect for the other. See [6] for a very detailed discussion of responsiveness.

To test whether assessor opinions vary, each TREC-8 question was independently judged by three different assessors. The separate judgments were combined into a single judgment set through adjudication for the official track evaluation, but the individual judgments were used to measure the effect of differences in judgments on systems' scores. Assessors opinions did vary. For example, assessors differed on how much of a name was required and on the desired granularity of dates and locations. Fortunately, as with document retrieval evaluation, the relative mean reciprocal rank scores between QA systems remain stable despite differences in the judgments used to evaluate them [5].

The TREC 2001 track modified the main task to make it more realistic and introduced two new tasks, the list task and the context task. All runs were restricted to answer strings of maximum length 50 bytes since the results from the earlier tracks clearly demonstrated that allowing 250-byte answer strings was a much simpler problem. In the main task, the guarantee that a question had an answer in the document collection was eliminated. A system returned the string "NIL" to indicate its belief that there was no answer in the document collection. NIL was marked correct if there was no known answer for that question in the collection and incorrect otherwise.

The list task required systems to assemble an answer from information located in multiple documents. Such questions are harder to answer than the questions used in the main task since information duplicated in the documents must be detected and reported only once. Each question in the list task specified a particular kind of information to be retrieved, such as *Who are 6 actors who have played Tevye in "Fiddler on the Roof"?*. Systems returned an unordered list of [*document-id*, *answer-string*] pairs where each pair represented a single instance. Results were scored using mean accuracy, which is the ratio of the number of distinct correct responses retrieved to the target number of responses requested.

The context task was a pilot evaluation for question answering within a particular scenario or context. The task was designed to represent the kind of dialog processing that a system would need to support an interactive user session. Questions were grouped into different series, and the QA system was expected to track the discourse objects across the individual questions of a series. Unfortunately, the results in the pilot were completely dominated by whether or not a system could answer the particular type of question: the ability to correctly answer questions later in a series was uncorrelated with the ability to correctly answer questions earlier in the series. Thus the task was not repeated in TREC 2002.

2 The TREC 2002 QA Track

The TREC 2002 track repeated the main and list tasks from 2001, but with the major difference of requiring systems to return exact answers. The change to exact answers was motivated by the belief that a system's ability to recognize the precise extent of the answer is crucial to improving question answering technology. The problems with using text snippets containing the answer as responses were illustrated in the TREC 2001 track. For example, each of the answer strings shown in Figure 1 was judged correct for the question *What river in the US is known as the Big Muddy?*, yet earlier responses are clearly better than later ones. Judging only exact answers correct forces systems to demonstrate that they know precisely where the answer lies in such strings.

What constitutes an "exact answer"? As with correctness, exactness is essentially a personal opinion. NIST provided guidelines to the assessors so that questions would be judged similarly, but in the end whether or not an answer was exact was up to the assessor. The guidelines given to the assessors are reproduced in Figure 2. Notice that even "good" responses that contain a correct answer and justification for that answer

```
the Mississippi
Known as Big Muddy, the Mississippi is the longest
as Big Muddy , the Mississippi is the longest
messed with .  Known as Big Muddy , the Mississip
Mississippi is the longest river in the US
the Mississippi is the longest river in the US,
the Mississippi is the longest river(Mississippi)
has brought the Mississippi to its lowest
ipes.In Life on the Mississippi,Mark Twain wrote t
Southeast;Mississippi;Mark Twain;officials began
Known; Mississippi; US,; Minnesota; Gulf Mexico
Mud Island,;Mississippi;"The;-- history,;Memphis
```

Figure 1: Correct answer strings for *What river in the US is known as the Big Muddy?*

were considered inexact for the purposes of this evaluation.

A system response consisting of an [*document-id, answer-string*] pair was assigned exactly one judgment by a human assessor as follows:

wrong: the answer string does not contain a correct answer or the answer is not responsive;

not supported: the answer string contains a correct answer but the document returned does not support that answer;

not exact: the answer string contains a correct answer and the document supports that answer, but the string contains more than just the answer (or is missing bits of the answer);

right: the answer string consists of exactly a correct answer and that answer is supported by the document returned.

Both QA tasks used the same new document collection as the source of answers. The collection is known as the AQUAINT Corpus of English News Text, which may be obtained from the Linguistic Data Consortium (`www.ldc.upenn.edu`) as catalog number LDC2002T31. The collection is comprised of documents from three different sources: the AP newswire from 1998–2000, the New York Times newswire from 1998–2000, and the (English portion of the) Xinhua News Agency from 1996–2000. There are approximately 1,033,000 documents and 3 gigabytes of text in the collection.

As in previous tracks, NIST provided the ranking of the top 1000 documents retrieved by the PRISE search engine when using the question as a query, and the full text of the top 50 documents per question (as given from that same ranking). This data was provided strictly as a convenience for groups that did not wish to implement their own document retrieval system. There was no guarantee that the ranking would contain the documents that actually answer a question.

All runs submitted to the track were required to be completely automatic; no manual intervention of any kind was permitted. To avoid any confounding effects caused by processing questions in different orders, all questions were required to be processed from the same initial state. That is, the system was not permitted to adapt to test questions that had already been processed.

Thirty-four groups submitted a total of 76 runs to the QA track, 67 main task runs (though two of the runs were mistakenly duplicates of one another) and 9 list task runs. All submissions to the track were judged.

3 The Main Task

In addition to the requirement for exact answers, the TREC 2002 main task had another significant change from earlier QA tasks. Systems were limited to one response per question, not five, and thus the scoring

Consider the question

What is the longest river in the United States?

The following are correct, exact answers

- `Mississippi,`
- `the Mississippi,`
- `the Mississippi River,`
- `Mississippi River`
- `mississippi`

while none of the following are correct, exact answers

- `At 2,348 miles the Mississippi River is the longest river in the US.`
- `2,348 miles; Mississippi`
- `Missipp`
- `Missouri`

You are not required to accept only the most minimal response possible as an exact answer; some redundancy is fine. In the Mississippi River example, we want you to accept "`Mississippi River`" as exact even though "river" is redundant since the correct response must be a river. Similarly, we want you to accept answers of "`<number> X`" for questions that ask "*How many X?*". Ungrammatical responses are probably not exact. A location question can have "`in Pennsylvania`" as an exact answer, but not "`Pennsylvania in`". If the answer string contains several entities of the same type, one of which is correct and the others are not, then the answer string is not exact.

Figure 2: Instructions given to the assessors regarding how to judge exact answers.

metric also changed. The scoring metric used, called the confidence-weighted score, was specifically chosen to test a system's ability to recognize when it has found a correct answer.

This section describes the main task within the TREC 2002 QA track. The first subsection gives the details regarding how the task was implemented. The following subsection provides the results of the evaluation, and the final subsection assesses the quality of the evaluation.

3.1 Task details

As already mentioned, one goal of the main task in this year's QA track was to test a system's ability to recognize when it has found a correct answer. A main task run consisted of exactly one response for each of 500 test questions. A response was either a [*document-id*, *answer-string*] pair or the string "NIL", which was used to indicate the system's belief that there was no correct answer in the collection. Within the submission file, the questions were ordered from most confident response to least confident response. That is, the question for which the system was most confident that it had returned a correct response was ranked first, then the question that the system was next most confident about, etc. so that the last question was the question for which the system was least confident in its response.

Given a question ranking based on confidence of a correct response, an analog of document retrieval's uninterpolated average precision can be computed. This measure rewards a system for a correct answer early in the ranking more than it rewards for a correct answer later in the ranking. More formally, if there are Q questions in the test set, the confidence-weighted score is defined to be

$$\frac{1}{Q}\sum_{i=1}^{Q}\frac{\text{number correct in first } i \text{ ranks}}{i}.$$

The test set of question used in the task were drawn from MSNSearch and AskJeeves logs. These logs are part of the set of logs donated by Microsoft and AskJeeves for use in TREC 2001. As in TREC 2001, NIST assessors searched the document collection for answers to candidate questions. NIST staff selected the final test set from among the candidates that had answers, keeping some questions for which the assessors found no answer. After judging, 46 questions had no known answer in the collection. NIST corrected the spelling, punctuation, and grammar of the questions in the logs, but left the content as it was. Unfortunately, some errors in the test questions remained. For example, question 1445 asked *When is Snoop Dog's birthday?*, when the correct spelling of the name is 'Snoop Dogg'. After discussion on the track mailing list, the track participants decided to evaluate over all 500 questions despite the remaining errors. This decision was based on the difficulty of knowing when to stop calling something an error (is a misplaced apostrophe an error? missing capitalization? alternate spellings?) and on the recognition that deployed systems will have to cope with user errors.

About one quarter of the 2001 test set of questions were definition questions such as *Who is Duke Ellington?* and *What are polymers?*. Having many definition questions is a problem for an evaluation such as TREC where there is no specific target user and thus no way of knowing what kind of response should be produced. Accordingly, NIST did not choose any definition questions for this year's test set, a restriction that most likely made the test set intrinsically easier than last year's set since definition questions are among the more difficult questions to answer. NIST made no other attempt to control the relative number of different types of questions in the test set.

One of the concerns expressed by the track participants regarding the move to exact answers was that submissions containing only exact answers would be too sparse to make good training data for future development. To address this concern and to collect data for a future "answer justification" task, participants were requested to also submit a justification for each of their responses. A justification was defined to be an arbitrary collection of ASCII characters that did not contain newline characters and was no longer than 1024 characters. Justifications were optional in that the justification string could be empty. The vast majority of justifications that were submitted consisted of the piece of text (snippet, sentence, or paragraph) from which the response had been extracted.

3.2 Evaluation results

Table 1 gives evaluation results for a subset of the main task runs. The table includes one run each from the fifteen groups who submitted the top-scoring runs. The run shown in the table is the run with the best confidence-weighted score, and the table is sorted by confidence-weighted score. Also given in the table are the number and percentage of questions answered correctly; the number of questions whose response was judged as inexact; and the precision and recall for recognizing when there is no correct answer in the document collection ("NIL Accuracy"). Precision of recognizing no answer is the ratio of the number of times NIL was returned and correct to the number of times it was returned; recall is the ratio of the number of times NIL was returned and correct to the number of times it was correct (46).

QA systems have become increasingly complex over the four years of the TREC track such that there is now little in common across all systems. Most systems classify an incoming question according to a system-specific ontology of question types as a first step. The ontologies of question types range from small sets of broad categories to highly-detailed hierarchical schemes. Once a question is classified, the system performs type-specific processing. Many TREC 2002 systems used specifc data sources such as name lists, gazetteers, movie databases, and the like that were searched when the system determined the question to be of the appropriate type. The web was used as a data source by most systems, though it was used in different ways. Some systems used the web as the primary source of an answer that the system then mapped to a document in the corpus to return as a response. Other systems did the reverse: used the corpus as the primary source of answers and then verified candidate answers on the web. Still other systems used the web as one of several sources whose combined evidence selected the final response.

TREC 2001 saw an increase in the number of systems using a shallow, data-driven approach to question answering in contrast to systems that attempt a full understanding of the question. Both approaches were well-represented in TREC 2002, as illustrated by the two top-scoring runs *LCCmain2002* from Language Computer Corporation and *exactanswer* from InsightSoft-M. The LCC system, PowerAnswer, transforms questions and possible answer text into a logic representation and then builds a formal proof for valid answers [1]. In contrast, the InsightSoft-M system relies on an extensive set of patterns where an individual

Table 1: Evaluation scores for a subset of the TREC 2002 main task runs. Scores are given for the best run as measured by confidence-weighted score from the top 15 groups. Scores include the confidence-weighted score; the number (#) and percentage (%) of questions that were answered correctly; the number of responses judged inexact; and the precision (Prec) and recall (Recall) for recognizing when there is no correct answer in the collection (NIL Accuracy).

	Confidence	Correct				
	weighted	Answers		Number	NIL Accuracy	
Run Tag	Score	#	%	Inexact	Prec	Recall
LCCmain2002	0.856	415	83.0	8	0.578	0.804
exactanswer	0.691	271	54.2	12	0.222	0.848
pris2002	0.610	290	58.0	17	0.241	0.891
IRST02D1	0.589	192	38.4	17	0.167	0.217
IBMPQSQACYC	0.588	179	35.8	9	0.196	0.630
uwmtB3	0.512	184	36.8	20	0.000	0.000
BBN2002C	0.499	142	28.4	18	0.182	0.087
isi02	0.498	149	29.8	15	0.385	0.109
limsiQalir2	0.497	133	26.6	11	0.188	0.196
ali2002b	0.496	181	36.2	15	0.156	0.848
ibmsqa02c	0.455	145	29.0	44	0.224	0.239
FDUT11QA1	0.434	124	24.8	6	0.139	0.957
aranea02a	0.433	152	30.4	36	0.235	0.174
nuslamp2002	0.396	105	21.0	17	0.000	0.000
pqas22	0.358	133	26.6	11	0.145	0.674

pattern is a complex expression built from simpler component structures [2].

The results in Table 1 illustrate that the quality of a system's confidence ranking can have a significant impact on its score. For example, the FDUT11QA1 and aranea02a runs have nearly identical confidence-weighted scores of .434 and .433, but the aranea02a run correctly answered 28 more questions than the FDUT11QA1 run. Figure 3 shows a scatter plot of confidence-weighted score versus number correctly answered for all main task runs. The solid line shows the best possible score a run could achieve for a given number of correctly answered questions; this score corresponds to ranking all correctly answered questions before all incorrectly answered questions. Similarly, the dotted line shows the worst possible score a run could achieve; this score corresponds to ranking all incorrectly answered questions before all correctly answered questions. In general, points are closer to the optimal line than the pessimal line, demonstrating that the systems were at least as good at ranking their responses as random guessing. A dot above and to the left of a second dot represents a system that is better at ranking than the second system since it has a higher confidence-weighted score but correctly answered fewer questions.

The systems used a variety of approaches to creating their question rankings. Almost all systems used question type as a factor since some question types are easier to answer than others. Some systems use a score to rank candidate answers for a question; when that score is comparable across questions, it can also be used to rank questions. A few groups used a training set of previous years' questions and answers to learn a good feature set and corresponding weights to predict confidence. Many systems used NIL as an indicator that the system couldn't find an answer (rather than the system was sure there was no answer), so ranked NIL responses last. With the exception of the *LCCmain2002* run, though, the NIL accuracy scores are low, indicating that systems had trouble recognizing when there was no answer in the document collection.

3.3 Analysis of the evaluation

The TREC-8 track demonstrated that QA evaluation results based on text snippets and mean reciprocal rank scoring is stable despite differences in assessor opinions as to whether an answer is correct [5]. This year's main task included several possible sources of additional instability: a single response per question,

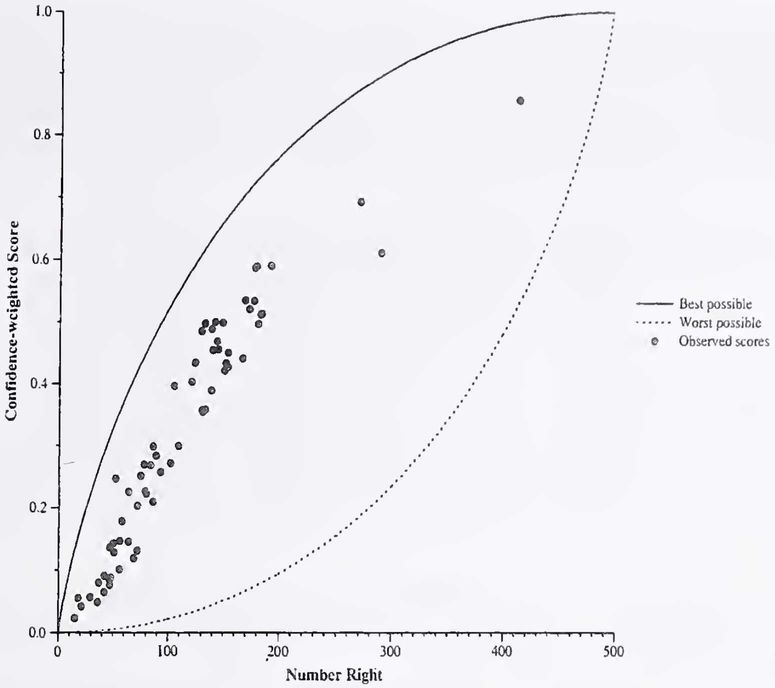


Figure 3: Confidence-weighted score vs. number correctly answered questions for main task runs.

Table 2: Kendall tau correlations for system rankings based on different judgment sets.

	Confidence score			Number correct		
	Set 1	Set 2	Set 3	Set 1	Set 2	Set 3
Adjudicated	0.954	0.941	0.944	0.958	0.949	0.960
Set 1		0.920	0.917		0.933	0.944
Set 2			0.906			0.926

confidence-weighting scoring, and exact answers. The methodology used in TREC-8 to test for stability was repeated for this year's main task to assess the effect of these changes. Each question was independently judged by three different assessors. The assessors for a particular question were arbitrarily assigned as assessor 1, assessor 2, or assessor 3. All the assessor 1 judgments for all questions were gathered into judgment set 1, all the assessor 2 judgments into judgment set 2, and all the assessor 3 judgments into judgment set 3. These three judgment sets were combined through adjudication into a final judgment set, which is the judgment set used to produce the official TREC 2002 main task scores.

Each run was scored using each of the four judgment sets. For each judgment set, the runs were ranked in order from most effective to least effective using either the confidence-weighted score or the raw number of correctly answered questions. The distance between two rankings was computed using a correlation measure based on Kendall's tau [3]. Kendall's tau computes the distance between two rankings as the minimum number of pairwise adjacent swaps to turn one ranking into the other. The distance is normalized by the number of items being ranked such that two identical rankings produce a correlation of 1.0, the correlation between a ranking and its perfect inverse is -1.0, and the expected correlation of two rankings chosen at random is 0.0. Table 2 gives the correlations between all pairs of rankings for both evaluation metrics.

The correlations between rankings are all above 0.9, an acceptable level for the assessor effect. Correlations with the adjudicated judgment set are 0.94 or higher, a better level. The higher correlations with the adjudicated set are probably due to the lower incidence of judgment errors (i.e., just plain mistakes rather than differences of opinion) in an adjudicated set. Correlations are slightly higher for the raw count of the

Table 3: Distribution of disagreements in judgments. Each response pair was independently assigned a judgment of wrong (W), right (R), unsupported (U), or inexact (X) by three assessors. Each entry in the table gives the number (#) and percentage (%) of pairs assigned the given triple of judgments. For example, for 418 pairs or 22.2 % of the total disagreements, two assessors marked the pair right and the third marked it inexact.

	Counts			Counts	
Judgments	#	%	Judgments	#	%
WWR	174	9.2	WXX	86	4.6
WWU	151	8.0	RRU	141	7.5
WWX	141	7.5	RRX	418	22.2
WRR	167	8.9	RUU	87	4.6
WRU	32	1.7	RUX	36	1.9
WRX	93	4.9	RXX	201	10.7
WUU	81	4.3	UUX	23	1.2
WUX	34	1.8	UXX	21	1.1

number of questions correctly answered than for the confidence-weighted score. This likely reflects the fact that the confidence-weighted score is much more sensitive to differences in judgments for questions at small (close to one) ranks.

There was a total of 15,948 [*document-id*, *answer-string*] pairs judged across the 500 questions, an average pool size of 31.9 strings. This is a smaller pool size than in previous tracks because only one response per question per run was allowed, and because the move to exact answers significantly increased the amount of overlap among runs. A total of 1886 pairs (11.8 % of all pairs) had some disagreement among the three assessors as to which judgment should be assigned to the pair. This small percentage of disagreement is misleading, however, since only 3725 pairs had at least one judge assign a judgment that was something other than wrong. In other words, there was some disagreement in assessing for half of all pairs that were not obviously wrong.

Table 3 shows the distribution of the assessors' disagreements. Each response pair is associated with a triple of judgments according to the three judgments assigned by the different assessors. In the table the judgments are denoted by W for wrong, R for right, U for unsupported, and X for inexact. The table shows the number of pairs that are associated with each triple plus the percentage of the total number of disagreements that that triple represents. The largest number of disagreements involves right and inexact judgments: the RRX and RXX combinations account for a third of the total disagreements. Inspection of these disagreements reveals that many of the granularity differences observed in TREC-8 are now reflected in this distinction. For example, question 1395 asks *Who is Tom Cruise married to?*, and a correct response is Nicole Kidman. Some assessors accepted just Kidman, but others marked that as inexact. Some assessors also accepted actress Nicole Kidman, while others marked that as inexact. Similar issues arose with dates and place names. For dates and quantities, there was disagreement whether slightly off responses are wrong or inexact. For example, when the correct response is April 20, 1999, is April 19, 1999 wrong or inexact? This last distinction doesn't matter very much in practice since in either case the response is not right.

Human judgments are not the only source of variability when evaluating QA systems. As is true with document retrieval systems, QA system effectiveness depends on the questions that are asked, so the particular set of questions included in a test set will affect evaluation results. Since the test set of questions is assumed to be a random sample of the universe of possible questions, there is always some chance that a comparison of two systems using any given test set will lead to the wrong conclusion. The probability of an error can be made arbitrarily small by using arbitrarily many questions, but there are practical limits to the number of questions that can be included in an evaluation.

Following our work for document retrieval evaluation [4], we can use the runs submitted to the QA track to empirically determine the relationship between the number of questions in a test set, the observed difference in scores (δ), and the likelihood that a single comparison of two QA runs leads to the correct conclusion. Once established, the relationship can be used to derive the minimum difference in scores required for a certain level of confidence in the results given there are 500 questions in the test set.

The core of the procedure is comparing the effectiveness of a pair runs on two disjoint question sets of equal size to see if the two sets disagree as to which of the runs is better. We define the error rate as the percentage of comparisons that result in a disagreement (a "swap"). Since the QA track used 500 questions, we can directly compute the error rate for question set sizes up to 250 questions. By fitting curves to the values observed for question set sizes up to 250, we can extrapolate the error rates to question sets up to 500 questions.

When calculating the error rate, the difference between two runs' confidence-weighted scores is categorized into one of 21 bins based on the size of the difference. The first bin contains runs with a difference of less than 0.01 (including no difference at all). The next bin contains runs whose difference is at least 0.01 but less than 0.02. The limits for the remaining bins increase by increments of 0.01, with the last bin containing all runs with a difference of at least 0.2.

Each question set size from 1 to 250 is treated as a separate experiment. Within an experiment, we randomly select two disjoint sets of questions of the required size. We compute the confidence-weighted score over both question sets for all runs, then count the number of times we see a swap for all pairs of runs using the bins to segregate the counts by size of the difference in scores. The entire procedure is repeated 10 times (i.e., we perform 10 trials), with the counts of the number of swaps kept as running totals over all trials.

The error rates computed from this procedure are then used to fit curves of the form $ErrorRate = A_1 e^{-A_2 S}$ where A_1 and A_2 are parameters to be estimated and S is the size of the question set. A different curve is fit for each different bin. The input to the curve-fitting procedure used only question set sizes greater than 20 since smaller question set sizes are both uninteresting and very noisy. Curves could not be fit for the first bin (differences less than .01), for the same reason, or for bins where differences were greater than 0.16. Curves could not be fit for large differences because too much of the curve is in the long flat tail.

The resulting extrapolated error rate curves are plotted in Figure 4. In the figure, the question set size is plotted on the x-axis and the error rate is plotted on the y-axis. The error rate for 500 questions when a difference of 0.05 in confidence-weighted scores is observed is approximately 8 %. That is, if we know nothing about two systems except their scores which differ by 0.05, and if we repeat the experiment on 100 different sets of 500 questions, then on average we can expect 8 out of those 100 sets to favor one system while the remaining 92 to favor the other.

The horizontal line in the graph in Figure 4 is drawn at an error rate of 5 %, a level of confidence commonly used in experimental designs. For question set sizes of 500 questions, there needs to be an absolute difference of at least 0.07 in confidence-weighted scores before the error rate is less than 5 %. When using the 5 % error rate standard, many of the runs in Table 1 group into classes that should be considered equally effective. For example, the scores for the *pris2002*, *IRST02D1*, and *IBMPQSQACYC* runs are all within 0.07 of one another. Further, all changes in the system rankings when systems were evaluated using different judgment sets were between pairs of systems whose difference in confidence-weighted scores when evaluated using the adjudicated judgment set was less than 0.07.

4 The List Task

The list task requires systems to assemble an answer from information located in multiple documents. Each question in the list task specified the number of instances of a particular kind of information to be retrieved, such as in the example questions shown in Figure 5. Each instance was guaranteed to obey the same constraints as an individual answer in the main task except that there was known to be at least as many correct instances as requested in the document collection. Systems returned an unordered list of [*document-id, answer-string*] pairs where each pair represented a single instance. The list could contain no more than the target number of instances.

The 25 questions used as the list task test set were constructed by NIST assessors. The assessors were instructed to construct questions whose answers would be a list of entities (people, places, dates, numbers) such that the list would not likely be found in a reference work such as a gazetteer or almanac. Each assessor was asked to create one small question (five or fewer expected answers), one large question (between twenty and forty expected answers), and two medium questions (between five and twenty expected answers). They

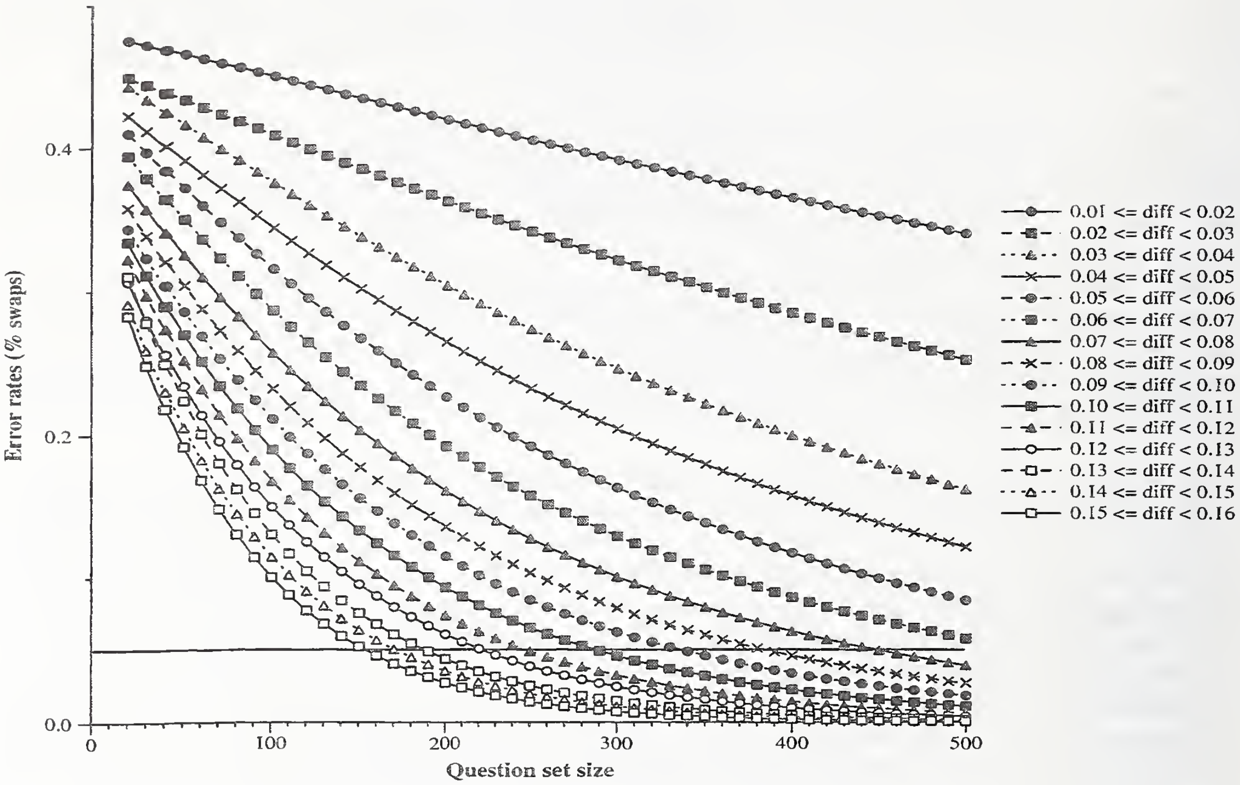


Figure 4: Error rates extrapolated to test sets of 500 questions.

- Name 22 cities that have a subway system.
- What are 5 books written by Mary Higgens Clark?
- List 13 countries that export lobster.
- What are 12 types of clams?
- Name 21 Godzilla movies.

Figure 5: Example list task questions.

Table 4: Average accuracy for list task runs. Accuracy is computed as the number of distinct instances divided by the target number of instances.

Run Tag	Average Accuracy	Run Tag	Average Accuracy
LCClist2002	0.65	shefft11lo	0.06
SUT11IR1LT2	0.15	sheft11log	0.06
SUT11IR1LT	0.11	clr02l1	0.06
UdeMlistNoW	0.07	clr02l2	0.05

searched the document collection using the PRISE search engine to find as complete a list of instances as possible. The target number of instances to retrieve was then selected such that the document collection contained more than the requested number of instances, but more than one document was required to meet the target. A single document could contain multiple instances, and the same instance might be repeated in multiple documents.

Judgments of incorrect, not supported, inexact, or right were made individually for each [*document-id, answer-string*] pair as in the main task. The assessor was given one list at a time, and while judging for correctness he also marked a set of responses as distinct. The assessor arbitrarily chose any one of a set of equivalent responses to mark as the distinct one, and marked the remainder as not distinct. Only correct responses could be marked distinct. Each question was judged by only one assessor, though the judgments were reviewed for errors by NIST staff.

List results were evaluated using accuracy, the number of distinct, correct responses divided by the target number of instances. Table 4 gives the average accuracy scores for the eight list task submissions.

In general, the scores for the list task are low. With the change to exact answers, retrieving distinct answers was not an issue: there was only one case across all questions and runs where two correct instances were deemed equivalent. The requirement for exact answers does not appear to be a major problem either. Of the 256 total instances requested across the 25 questions, the eight runs averaged only 7.1 "inexact" judgments. The average for unsupported judgments was similar at 7.9. Instead, it appears that the target answers were just difficult to find.

5 Summary

The TREC 2002 QA track made significant changes to the task definition as compared to earlier TREC tracks. In particular, the 2002 main task required systems to return exact answers, to return only one response per question, and to rank questions by confidence in the response. Major themes for this year's systems were a marked increase in the use of name lists, gazetteers, and the like to answer specific question types, and continued reliance on the web as a system component.

Evaluation of the track results confirmed that system comparisons are sufficiently stable for an effective evaluation. Human assessors do not always agree as to whether an answer is exact, but the differences reflect the well-known differences in opinion as to correctness rather than inherent difficulty in recognizing whether an answer is exact. Empirically-derived error rates based on the sensitivity of the confidence-weighted score to different question sets suggest that scores differing by less than 0.07 are equivalently effective. No pair of systems with a difference in scores of at least 0.07 swapped when evaluated by different judgment sets.

Participation in the list task was quite limited. A majority of main task participants indicated that they did not perform the list task because of time constraints rather than a lack of interest in the task. Accordingly, the current plans for the TREC 2003 QA track are to have one task in which systems are required to answer a variety of question types, including factoid questions, list questions, and definition questions. The test question set will not explicitly distinguish the type of the question. During the evaluation phase, the question set will be partitioned into the three types and each type of question will be scored using the methodology appropriate for that question type.

Acknowledgements

My thanks to John Prager who suggested the plot in Figure 3, and Chris Buckley for fitting the error rate curves shown in Figure 4.

References

[1] Dan Moldovan, Sanda Harabagiu, Roxana Girju, Paul Morarescu, Finley Lacatusu, Adrian Novischi, Adriana Badulescu, and Orest Bolohan. LCC tools for question answering. In Voorhees and Buckland [7].

[2] Martin M. Soubbotin and Sergei M. Soubbotin. Use of patterns for detection of answer strings: A systematic approach. In Voorhees and Buckland [7].

[3] Alan Stuart. Kendall's tau. In Samuel Kotz and Norman L. Johnson, editors, *Encyclopedia of Statistical Sciences*, volume 4, pages 367–369. John Wiley & Sons, 1983.

[4] Ellen M. Voorhees and Chris Buckley. The effect of topic set size on retrieval experiment error. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 316–323, 2002.

[5] Ellen M. Voorhees and Dawn M. Tice. Building a question answering test collection. In *Proceedings of the Twenty-Third Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 200–207, July 2000.

[6] Ellen M. Voorhees and Dawn M. Tice. The TREC-8 question answering track evaluation. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pages 83–105, 2000. NIST Special Publication 500-246. Electronic version available at `http://trec.nist.gov/pubs.html`.

[7] E.M. Voorhees and L.P. Buckland, editors. *Proceedings of the Eleventh Text REtrieval Conference (TREC 2002)*, 2003.

The TREC-2002 Video Track Report

Alan F. Smeaton {asmeaton@computing.dcu.ie}
Centre for Digital Video Processing
Dublin City University
Glasnevin, Dublin 9, Ireland

Paul Over {over@nist.gov}
Retrieval Group
Information Access Division
National Institute of Standards and Technology
Gaithersburg, MD 20899-8940, USA

March 5, 2003

1 Introduction

TREC-2002 saw the second running of the Video Track, the goal of which was to promote progress in content-based retrieval from digital video via open, metrics-based evaluation. The track used 73.3 hours of publicly available digital video (in MPEG-1/VCD format) downloaded by the participants directly from the Internet Archive (Prelinger Archives) (internetarchive, 2002) and some from the Open Video Project (Marchionini, 2001). The material comprised advertising, educational, industrial, and amateur films produced between the 1930's and the 1970's by corporations, nonprofit organizations, trade associations, community and interest groups, educational institutions, and individuals. 17 teams representing 5 companies and 12 universities — 4 from Asia, 9 from Europe, and 4 from the US — participated in one or more of three tasks in the 2001 video track: shot boundary determination, feature extraction, and search (manual or interactive). Results were scored by NIST using manually created truth data for shot boundary determination and manual assessment of feature extraction and search results.

This paper is an introduction to, and an overview of, the track framework — the tasks, data, and measures — the approaches taken by the participating groups, the results, and issues regrading the evaluation. For detailed information about the approaches and results, the reader should see the various site reports in the final workshop proceedings.

1.1 New in TREC 2002

At the TREC 2001 video track workshop in November 2001, the track set a number of goals for improvement (Smeaton, Over, & Taban, 2002) and in the subsequent months through cooperative effort met almost all of them. As a result the 2002 track differs from the first running in 2001 in a number of important ways itemized here:

- There was an increase in the number of participants, up to 17 from last year's 12, and an increase in the data where a total of overview 73 hours of VCD/MPEG-1 data were identified for use in development and testing — up from 11 hours last year.
- A semantic feature extraction task was added. 10 features (e.g., cityscape, face, instrumental sound, monologue speech) were defined by a group of interested track participants and systems attempted with some success to find shots containing a given feature.
- Several groups volunteered to extract sets of these features from the test video and share their results with other groups allowing those other groups to use that feature detection in the search task. These feature detections were distributed in an MPEG-7 format developed by IBM.
- This year the track used a common set of shot definitions, donated by the CLIPS-IMAG group and formatted by Dublin City University whereas previously each group had defined their

own shot boundaries. Results for the feature detection and search tasks were reported in terms of these predefined units — allowing for pooling of results.

- The 25 topics for the search task were developed by NIST rather than by the participants and were released 4 weeks before the search results were due. These were again true multimedia queries as they all had video clips, images, or audio clips as part of the query, in addition to a text description. They reflect many of the various sorts of queries real users pose: requests for video with specific people or types of people, specific objects or instances of object types, specific activities or locations or instances of activity or location types (Enser & Sandom, 2002). Unlike last year, where the topics were either known item or general, this year's topics were all general.

- The very difficult task of fully automatic topic-to-query translation was set aside for a future TREC video track. Searching in this year's track could be interactive with full human access to multiple interim search results, or "manual". In manual searches a human with knowledge of the query interface but no direct or indirect knowledge of the search test set or search results was given one chance to translate each topic to what he or she believed to be the most effective query for the system being tested.

- The shot boundary detection test set was not announced until 3 weeks before the submissions were due at NIST for evaluation. New and revised measures were used to separate a system's ability to detect shot transitions by identifying at least one of the frames in the transition from the accuracy with which a system locates the entire transition (frame-recall and frame-precision).

- Elapsed search time was added as measure of effort for the interactive search task and groups were encouraged to gather and report data on searcher characteristics and satisfaction.

Details about each of the three tasks follow.

2 Shot boundary detection

Movies on film stock are composed of a series of still pictures (frames) which, when projected together rapidly, the human brain smears together so we get the illusion of motion or change. Digital video is also organized into frames - usually 25 or 30 per second. Above the frame, the next largest unit of video both syntactically and semantically is called the shot. A half hour of video, in a TV program for example, can contain several hundred shots. A shot was originally the film produced during a single run of a camera from the time it was turned on until it was turned off or a subsequence thereof as selected by a film editor. The new possibilities offered by digital video have blurred this definition somewhat, but shots, as perceived by a human, remain a basic unit of video, useful in a variety of ways.

Work on algorithms for automatically recognizing and characterizing shot boundaries has been going on for some time with good results for many sorts of data and especially for abrupt transitions between shots. Software has been developed and evaluations of various methods against the same test collection have been published e.g., using 33 minutes total from five feature films (Aigrain & Joly, 1994); 3.8 hours total from television entertainment programming, news, feature movies, commercials, and miscellaneous (Boreczky & Rowe, 1996); 21 minutes total from a variety of action, animation, comedy, commercial, drama, news, and sports video drawn from the Internet (Ford, 1999); an 8-hour collection of mixed TV broadcasts from an Irish TV station recorded in June, 1998 (Browne et al., 2000).

An open evaluation of shot boundary determination systems was designed by the OT10.3 Thematic Operation (Evaluation and Comparison of Video Shot Segmentation Methods) of the GT10 Working Group (Multimedia Indexing) of the ISIS Coordinated Research Project in 1999 using 2.9 hours total from eight television news, advertising, and series videos (Ruiloba, Joly, Marchand-Maillet, & Quénot, 1999).

The shot boundary task is included in the video track as an introductory problem, the output of which is needed for higher-level tasks such as search. Groups can participate for the first time on this task, develop their infrastructure, and move on to more complicated tasks the next year. Information on the effectiveness of particular systems is useful in selecting donated segmentations used for scoring other tasks.

2.1 Data

The shot boundary test collection for this year's TREC task comprises 4 hours and 51 minutes of video, slightly smaller than last year. The videos are mostly of a documentary/educational nature but were varied in their age, production style, and quality. There were 18 videos encoded in MPEG-1 with a total size of 2.88 gigabytes. The videos contained 545,068 total frames and 2,090 shot transitions (according to the manually created reference data.)

The reference data was created by a student at NIST whose task was to identify all transitions and assign each to one of the following categories:

cut - no transition, i.e., last frame of one shot followed immediately by the first frame of the next shot, with no fade or other combination;

dissolve - shot transition takes place as the first shot fades out *while* the second shot fades in

fadeout/in - shot transition takes place as the first shot fades out and *then* the second fades in

other - everything not in the previous categories e.g., diagonal wipes.

Software was developed and used to sanity check the manual results for consistency and some corrections were made.

The freely available software tool [1] was used to view the videos and frame numbers. The collection used for evaluation of shot boundary determination contains 2,090 transitions with the following breakdown as to type:

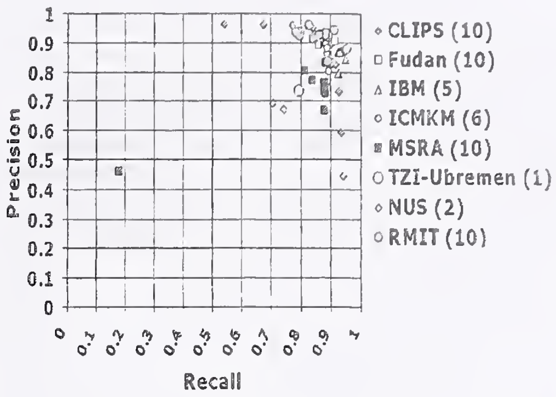
- 1466 — hard cuts (70.1%)
- 511 — dissolves (24.4%)
- 63 — fades to black and back (3.0%)
- 50 — other (2.4%)

Gradual transitions are generally harder to recognize than abrupt ones. The proportion of gradual transitions to hard cuts in this collection is about twice that reported by Boreczky and Rowe (1996) and by Ford (1999). This is due to the nature and genre of the video collection we used.

2.2 Evaluation

Participating groups in this task were allowed up to 10 submissions and these were compared automatically to the shot boundary reference data. Each group determined the different parameter settings for each run they submitted. Detection performance for cuts and for gradual transitions was measured by precision and recall where the detection criteria required only a single frame overlap between the submitted transitions and the reference transition. This was to make the detection independent of the accuracy of the detected boundaries. For the purposes of detection, we considered a submitted abrupt transition to include the last pre-transition and first post-transition frames so that it has an effective length of two frames (rather than zero).

Figure 1: Precision and recall for cuts



Analysis of performance individually for the many sorts of gradual transitions was left to the participants since the motivation for this varies greatly by application and system.

As last year, gradual transitions could only match gradual transitions and cuts match only cuts. except in the case of very short gradual transitions (5 frames or less), which, whether in the reference set or in a submission, were treated as cuts. We also expanded each abrupt reference transition by 5 frames in each direction before matching against submitted transitions to accommodate differences in frame numbering by different decoders.

Accuracy for reference gradual transitions successfully detected was measured using the one-to-one matching list output by the detection evaluation. The accuracy measures were frame-based precision and recall. Note that a system could be very good in detection and have poor accuracy, or it might miss a lot of transitions but still be very accurate on the ones it finds.

[1] The VirtualDub (Lee, 2001) website contains information about VirtualDub tool and the MPEG decoder it uses. The identification of any commercial product or trade name does not imply endorsement or recommendation by the National Institute of Standards and Technology.

Figure 2: Precision and recall for gradual transitions

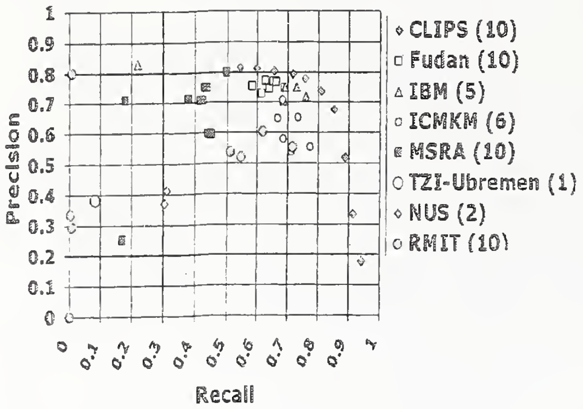
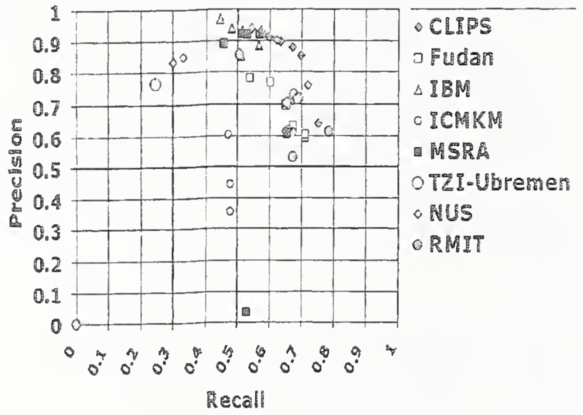


Figure 3: Frame-precision and frame-recall for gradual transitions



2.3 Results

As illustrated in Figure 1 and Figure 2, performance on gradual transitions lags, as expected, behind that on abrupt transitions, where for some uses the problem may be considered a solved one. The numbers in parentheses give the number of runs submitted by each group. Some groups (e.g., CLIPS and RMIT) used their runs to explore a number of precision-recall settings and seem to have good control of this tradeoff. Figure 3 indicates that at the level of frames in gradual transitions, the best systems have better precision than they do in detecting those transitions but their frame-level recall scores tend to be lower than for simple detection.

3 Feature extraction

A potentially important asset to help video search/navigation is the ability to automatically identify the occurrence of various semantic features such as "Indoor/Outdoor","People", "Speech" etc., which occur frequently in video information. The ability to detect features is an interesting challenge by itself but it would take on added importance if it could serve as an extensible basis for query formation and search. The high-level feature extraction task had the following objectives:

- to begin work on a benchmark for evaluating the effectiveness of detection methods for various semantic concepts
- to allow exchange of feature detection output based on the TREC Video Track search test set prior to the search task results submission date, so that a greater number of participants could explore innovative ways of leveraging those detectors in answering the search task queries.

The task was as follows. Given a standard set of shot boundaries for the feature extraction test collection and a list of feature definitions, participants were to return for each feature the list, at most the top 1000 video shots from the standard set, ranked according to the highest possibility of detecting the presence of the feature. The presence of each feature was assumed to be binary, i.e., it was either present or absent in the given standard video shot. If the feature was true for some frame (sequence) within the shot, then it was true for the shot. This is a simplification adopted for the benefits it afforded in pooling of results and approximating the basis for calculating recall.

The feature set was suggested in on-line discussions by track participants. The number of features to be detected was kept small so as to be manageable in this first implementation and the features were ones for which more than a few groups could create detectors. Another consideration was whether the features could, in theory at least,be used in executing searches on the video data using the topics. The topics did not exist yet at the time the features were defined. The feature definitions were to be in terms a human judge could understand.

Much to the appreciation of the track as a whole, some participating groups made their feature detection output available to participants in the search task and this will be discussed in the section describing the search task.

The features to be detected were defined as follows for the system developers and for the NIST assessors:

Outdoors segment contains a recognizably outdoor location, i.e., one outside of buildings. Should exclude all scenes that are indoors or are close-ups of objects (even if the objects are outdoors)

Indoors segment contains a recognizably indoor location, i.e., inside a building. Should exclude all scenes that are outdoors or are close-ups of objects (even if the objects are indoors).

Face segment contains at least one human face with the nose, mouth, and both eyes visible. Pictures of a face meeting the above conditions count.

People segment contains a group of two or more humans, each of which is at least partially visible and is recognizable as a human.

Cityscape segment contains a recognizably city/urban/suburban setting.

Landscape segment contains a predominantly natural inland setting, i.e., one with little or no evidence of development by humans. For example, scenes consisting mostly of plowed/planted fields, pastures, orchards would be excluded. Some buildings, if small features on the overall landscape, should be OK. Scenes with bodies of water that are clearly inland may be included.

Text Overlay segment contains superimposed text large enough to be read.

Speech a human voice uttering words is recognizable as such in this segment

Instrumental Sound sound produced by one or more musical instruments is recognizable as such in this segment. Included are percussion instruments.

Monologue segment contains an event in which a single person is at least partially visible and speaks for a long time without interruption by another speaker. Pauses are OK if short.

3.1 Data

23.26 hours (96 videos containing 7,891 standard shots) were randomly chosen from the total available data, to be used solely for the development of feature extractors. 5.02 hours (23 videos containing 1,848 standard shots) were randomly chosen from the remaining material for use as a feature extraction test set.

Table 1: Features and total hits

Feature name	Feature number	Shots submitted	Shots judged (pooled)	Total hits
Outdoors	1	12353	1821	962
Indoors	2	9143	1801	351
Face	3	7181	1688	415
People	4	4440	1233	486
Cityscape	5	9346	1656	521
Landscape	6	7208	1524	127
Text overlay	7	8120	1699	110
Speech	8	15800	1599	1382
Instrumental sound	9	11388	1846	1221
Monologue	10	5092	1319	38

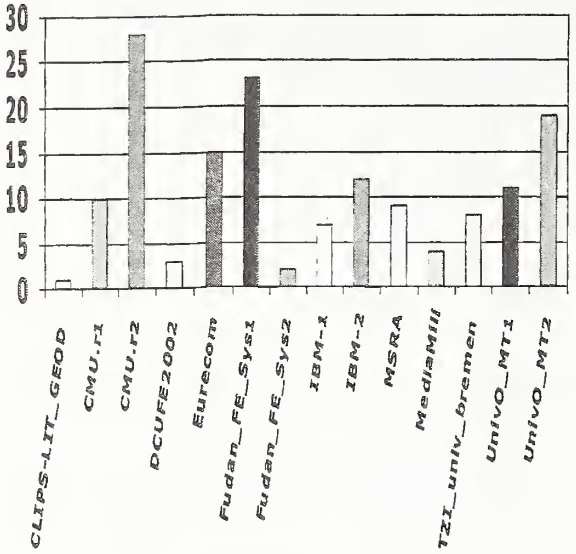
3.2 Evaluation

This year all result sets from all runs were fully assessed manually to create reference data. Basically, the feature extraction definitions were treated like topics of the form: "I want shots for which this feature is true."

3.3 Measures

The trec_eval software, a tool available via trec.nist.gov, was used to calculate recall, precision, average precision, etc., for each result. In experimental terms the features represent fixed rather than random factors, i.e., we were interested at this point in each feature rather than in the set of features as a random sample of some population of features. For this reason and because different groups worked on very different numbers of features, we did not aggregate measures at the run-level in the results pages at the back of the notebook. Comparison of systems should thus be "within feature".

Figure 4: The number of true shots contributed uniquely by run



3.4 Issues

It should be noted that in the case of some features (speech, instrumental sound) the number of shots in the feature extraction test set containing the feature approached or exceeded the maximum size of the submitted result set (1,000) and represented a large portion of the entire feature test collection size (1,848 shots) — see Table 1. While the performance of a random baseline was high in these cases, the median performance was still well above it. Where more hits exist than a result can hold, an artificial upper bound on possible average precision scores exists — namely for feature 8 (speech) 0.724 and for feature 9 (instrumental sound) 0.819.

3.5 Results

Figure 5 summarizes the results by feature for all of the runs at the median or above. Included as a dotted line in this figure is the baseline - the average for 100,000 randomly created result sets for each feature. The artificial upper limit on average precision mentioned above is indicated by a white triangle for features 8 and 9.

Results vary in their dispersion among features as well as in their mean. While the random baseline is high, almost all of the runs are well above it. While there was a lot of overlap in the shots submitted for a given feature, Figure 4 shows the relatively small number of true shots contributed uniquely by a given system – summed over all features. Not all systems submitted results for all features. The large overlap is no doubt due in part to the relatively small size of the test set (1,848 shots) in comparison to the size of the result (1,000 shots).

4 Search

The search task in the Video Track was an extension of its text-only analogue. Video search systems, all of which included a human in the loop, were presented with topics — formatted descriptions of an information need — and were asked to return a list of up to 100 shots from the videos in the search test collection which met the need. The list was to be prioritized based on likelihood of relevance.

4.1 Data to be searched

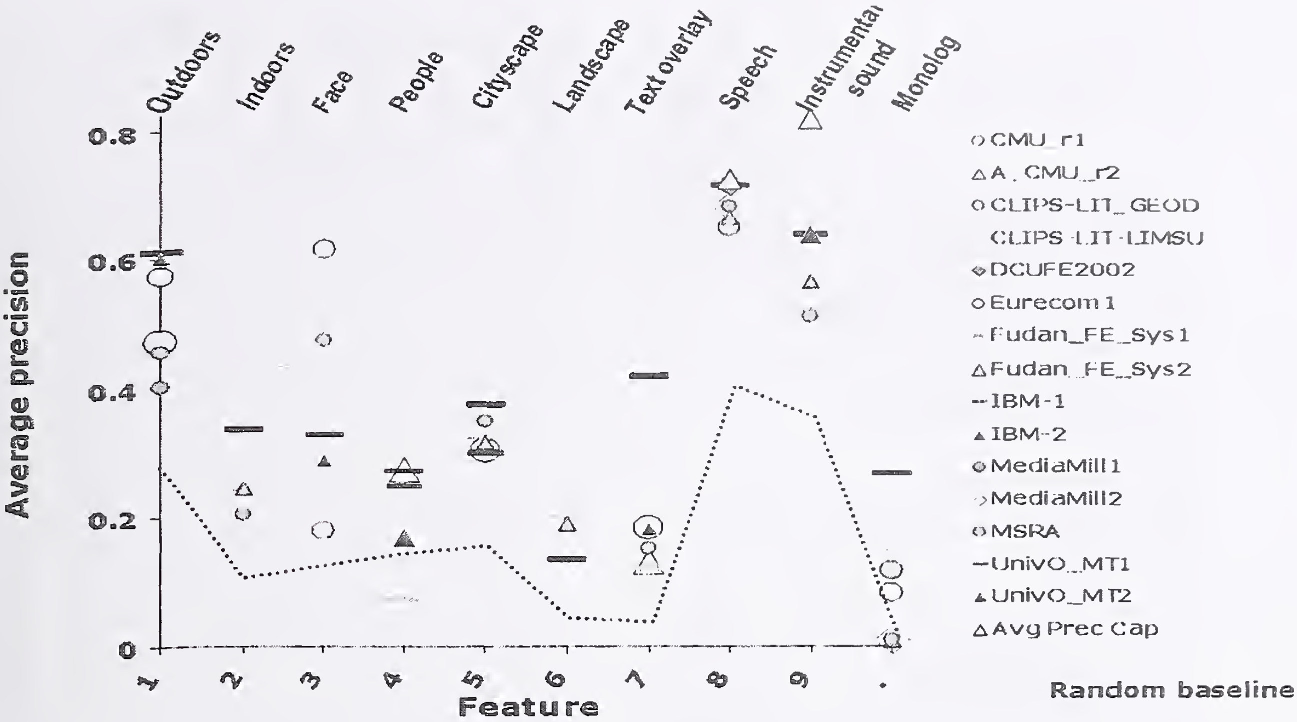
40.12 hours (176 videos containing 14,524 master shots) were randomly chosen from the identified collection to be used as the search test collection.

The video data was chosen because it represented an established archive of publicly available material that one can easily imagine being searched for information as well as historically interesting material that could be included in new video products. Publicly available video collections of any significant size are extremely hard to find. While we are not aware of any systematic study of the characteristics of the Internet Archive movie material, some details can be found in individual site papers. Collection characteristics will affect the scope of any conclusions drawn here.

Groups were allowed to develop their systems with knowledge of the search test collection — the topics being the surprise element. This was designated training pattern A. Other groups preferred to develop their systems without knowledge of the search test set. This training pattern was designated B. Results were labeled with these designations as were the feature extractions donated by some of the groups.

As was mentioned earlier, two search modes were allowed, fully interactive and manual, though no fully automatic mode was included, a choice which has advantages as well as disadvantages. A big problem in TREC video searching is that topics were complex and designating the intended meaning and interrelationships between the various pieces — text, images, video clips, and audio clips — is a complex one and the examples of video, audio, etc. do not always represent the information need exclusively and exhaus-

Figure 5: Average precision by feature and run



tively. Understanding what an image is of/about is famously complicated (Shatford, 1986).

The definition of the manual mode allowed a human, expert in the search system interface, to interpret the topic and create an optimal query in an attempt to make the problem less intractable. The cost of the manual mode is terms of allowing comparative evaluation is the conflation of searcher and system effects. However if a single searcher is used for all manual searches within a given research group, comparison of searches within that group is still possible. At this stage in the research, the ability of a team to compare variants of their system is arguably more important than the ability to compare across teams, where results are more likely to be confounded by other factors hard to control (e.g. different training resources, different low-level research emphases , etc.).

4.2 Topics

The topics were designed as multimedia descriptions of an information need, such as someone searching a large archive of video might have in the course of collecting material to include in a larger video or to answer questions. Today this may be done largely by searching descriptive text created by a human when the video material was added to the archive. The track's search scenario envisioned allowing the searcher to use a combination of other media in describing his or her need. How one might do this naturally and effectively is an open question. This year 25 topics were created by NIST, who had intended to create 50, but due to time pressures, this was not possible. Each topic contained a text description of the user information need. Examples in other media, e.g., one more video clips, still images, audio files illustrating the information need, were optional. Table 2 presents an overview of the topics, their types, and the number of relevant shots found for each topic.

Comparing the TREC video topic types to distributions of actual queries against video archives is nearly impossible due to lack of published studies, differences in archive content and searcher characteristics, amount of mediation, etc. However, Armitage and Enser (1996) provide some real world reference points which may be of interest. Comparing the distribution of TREC video track topics types to a sample of 370 submitted to the BBC Natural History Unit and 388 submitted to the British Film Institute's Na-

Table 2: Overview of topics

Topic #	Abbreviated text description of needed information/shot	Topic Types Panofsky-Shatford mode/facet categories (minus abstract) after Armitage & Enser 1996								Number of examples in the topic		Shots sub-mitted	Shots judged (pooling top 50 from each result)	Shots judged relevant
		Specific				Generic				Video	Image			
		S1	S2	S3	S4	G1	G2	G3	G4					
		person; group; tning	event; action	location	linear time; date; period	kind of person; thing	kind of event; action; condition	kind of place: geographical; architectural	cyclical time: season; time of day					
75	Eddie Rickenbacker	x								2	2	2668	850	15
76	Raymond H. Chandler	x								3	0	3036	625	47
77	pictures of George Washington	x								1	1	2521	931	3
78	depictions of Abraham Lincoln	x								1	1	2637	1014	6
79	people spending leisure time at the beach					x	x	x		4	0	3109	1055	55
80	one or more musicians					x	x			2	0	2829	860	63
81	football players					x				4	0	2311	890	15
82	women standing in long dresses					x	x			3	0	2696	1058	170
83	Golden Gate Bridge	x		x						0	5	2529	936	33
84	Price Tower in Bartlesville, OK	x		x						0	1	2409	816	4
85	Washington Square Park's arch in NYC	x		x						1	0	2708	909	7
86	overhead views of cities					x		x		4	0	3041	1112	105
87	oil fields, rigs, derricks					x		x		1	0	2721	1002	40
88	map of the continental US			x		x				4	0	2569	969	72
89	a living butterfly					x	x			0	2	2325	979	10
90	snow-capped mountain peaks or ridges					x		x		3	0	2785	926	75
91	one or more parrots					x				1	1	2228	880	17
92	sailboats, clipper ships, etc. with sails unfurled					x				4	2	2860	921	47
93	live beef or dairy cattle					x				5	0	3622	1003	161
94	groups of people walking in an urban environment					x	x	x		3	0	3168	1175	303
95	a nuclear explosion with a mushroom cloud					x	x			3	0	2658	951	17
96	one or more US flags flapping	x				x	x			2	0	2458	1055	31
97	microscopic views of living cells					x	x			2	0	2968	859	82
98	a locomotive approaching the viewer					x	x			5	0	2729	998	56
99	a rocket or missile taking off					x	x			2	0	2438	907	11

tional Film and Television Archive one sees the same predominance of non-abstract types and roughly the same percentage of type overlap (i.e., multi-category queries). However, the TREC queries have about half as many requests for specific persons and things and two to five times as many requests for generic persons and things. Whether this may be due to any degree to librarian/archivist mediation (e.g, substitution of a request for a known example for a generic request) is unknown.

4.3 Evaluation

The top 50 items (half) of each submitted result set was judged by a NIST assessor. Double judging in TREC 2001 indicated a high degree of assessor agreement for both relevant and non-relevant shots, so NIST did not do double judgments for TREC 2002.

4.4 Measures

The trec_eval program was used to calculate recall, precision, average precision, etc. The interested reader should see the back of the proceeding results pages for details on the performance of individual runs.

It should be noted that in the case of topics 82, 86, 93, and 94, as with evaluation in the feature extraction task, the number of relevant shots exceeded the maximum size of the submitted result set (100) — see Table 2. Where more relevant shots exist than a result can hold, an artificial upper bound on possible average precision scores exists — namely for topic 82 - 0.588, 86 - 0.952, 93 - 0.621, and 94 - 0.330.

4.5 Issues

Because the topics have a huge effect on the results, the topic creation process deserves special attention here. Ideally the topics would have been created by real users against the same collection used to test the systems, but such queries were not available.

Alternatively, interested parties familiar in a general way with the content covered by a test collection could have formulated questions which were then checked against the test collection to see that they were indeed relevant. This avenue was also not open to us for two main reasons. First, the collection used is so diverse that creating a question that has answers in several videos is next to impossible without detailed knowledge of the collection. Second, NIST had no video search system in place which could be used.

What was left was to work backward from the test collection with a number of goals in mind. Rather than attempt to create a representative sample, NIST tried to get an equal number of each of the basic types: generic/specific; person/thing/event, though in no way do we wish to suggest these types are equal as measured by difficulty to systems. Another important consideration was the estimated number of relevant shots and their distribution across the videos. The goals here were as follows:

- For almost all topics, there should be multiple shots that meet the need.
- If possible, relevant shots for a topic should come from more than one video.
- As the search task is already very difficult, we don't want to make the topics too difficult.

The videos in the test collection were viewed and notes made about their content in terms of people, things, and events, named or unnamed. Those that occurred in more than one video became candidates for topics. This process provided a rough idea of a minimum number of relevant shots for each candidate topic. The third goal was the most difficult since there is no reliable way to predict the hardness of a topic.

In general NIST tried to be sure there were relevant shots with relatively large images of the target person, thing, or event. When choosing examples for the topics, NIST tried to find at least some that seemed to resemble the target shot in shape, color, and/or texture. This was often not possible, nor is it likely the estimate of similarity corresponded in any meaningful way with that of the automatic systems.

Sometimes words from the audio were incorporated into the wording of the topic. This leaves open the possibility that some topics were in fact generally biased toward approaches using automatic speech recognition. On the other hand some information needs make demands unlikely to be supported by text from the audio e.g., requests for specific relative object/camera motion (98: locomotive approaching the viewer), some events/activities (96: US flags flapping), etc. A full analysis on the presence or absence of topic keywords in the audio track for relevant shots would be required to determine whether this is the case and has yet to be done.

The nature of the test collection for 2003 and the possible use of a search tool to validate minimal numbers of relevant shots (even if a related system is likely to participate in the evaluation), should allow the creation of topics uncontaminated by the details of the test collection.

Figure 6: Top 10 manual search runs

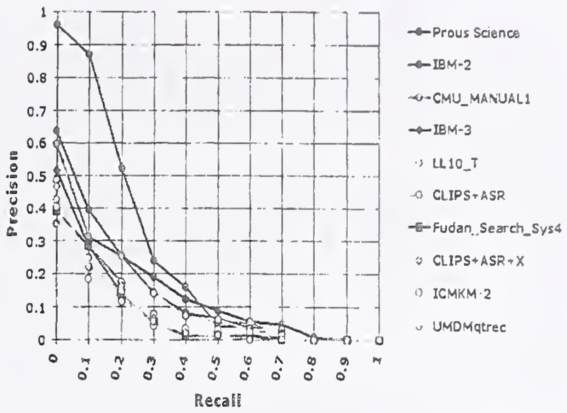
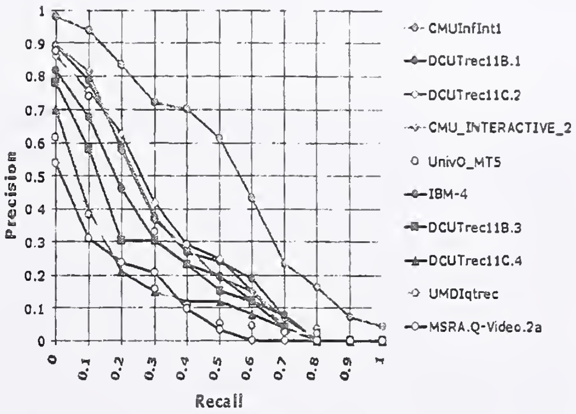


Figure 7: Top 10 interactive search runs

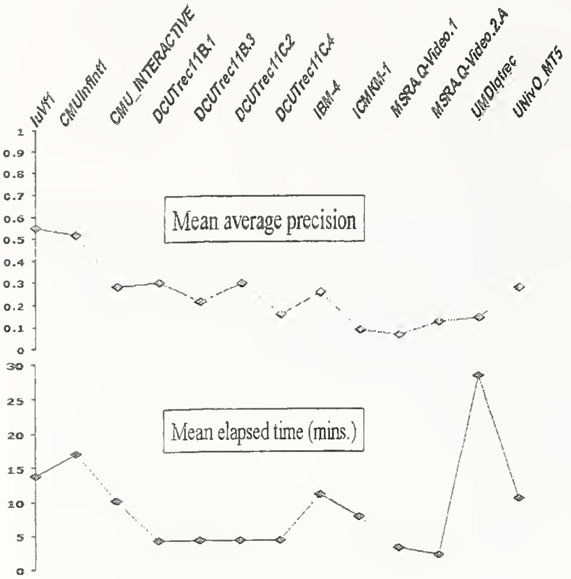


4.6 Results

The results in terms of mean average precision for the top ten manual runs are presented in Figure 6 and those for the top ten interactive runs in Figure 7, each list sorted by mean average precision. Another measure for interactive runs which was gathered was total elapsed time for each topic search. Figure 8 contrasts the two measures. Time spend varied widely from an average of just over 1 minute to just under 30 minutes per topic. No simple relationship between elapsed search time and effectiveness as measured by mean average precision is apparent.

The number of relevant shots contributed uniquely

Figure 8: MAP vs mean elapsed time



by each run is presented in Figure 9. As expected, interactive runs contribute the most.

The search task results in this report are based on manual relevance judgments for the top (most relevant) half (50 shots) in each submitted result set. The bottom half of each result has also been judged manually and this yielded few additional relevant shots except in the case of a couple topics which already had more than the average number of relevant shots. Fourteen of the twenty-five topics had no change in the number of relevant shots. For 8 the number of relevant shots grew 11% or less, for 3 it grew 20 - 24% (topics 82 - 20%; 94 - 21%; 96 - 24%). Figure 10 illustrates the distribution of relevant shots in the top versus the bottom half of the result sets.

Looking underneath the averages at the performance by topic, one can see that considerable variability exists across the set of topics and that some topics were harder than others. Figure 11 and Figure 12 show these together with two covariates: number of relevant shots and relevant videos. Manual results for topics 76, 84, 90, and 97 stand out. Why are they better? No single, simple explanation suffices. Topics with more relevant shots/videos or topics containing video examples from the search test collection (see small vertical arrows in Figure 12) are not necessarily easier.

The jury is still out with respect to two important search issues. The reliable usefulness of features in search generally or in specific situations has yet to be

Figure 9: Relevant shots contributed uniquely by run

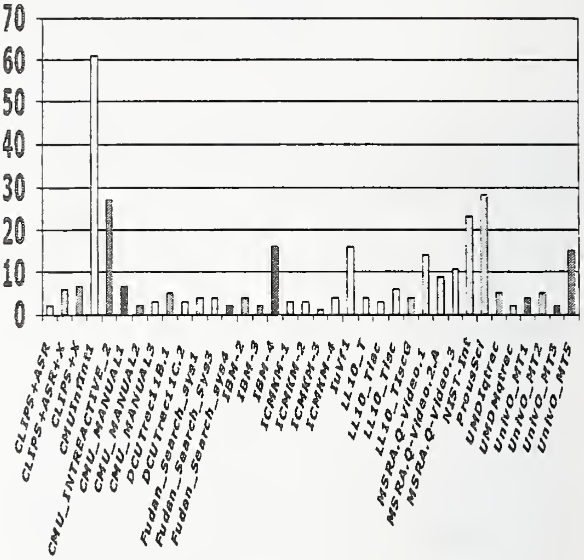


Figure 10: Distribution of relevant shots in result sets

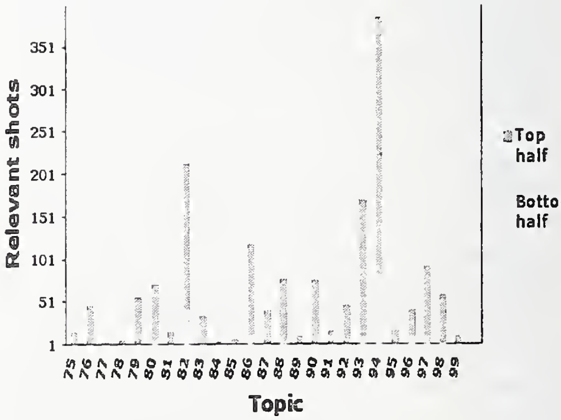


Figure 11: Interactive search: average precision by topic

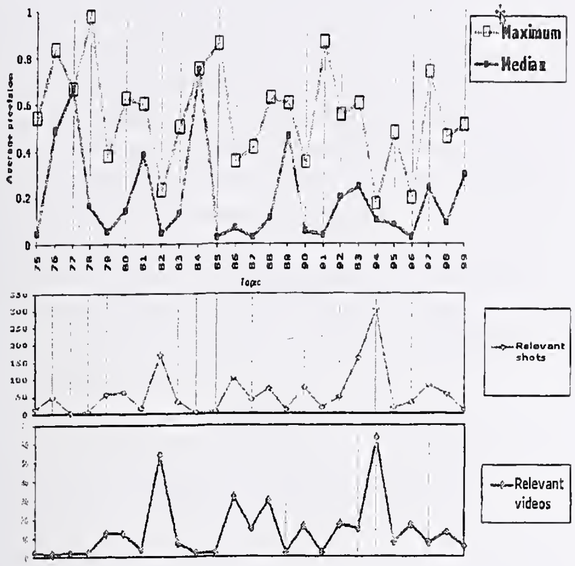
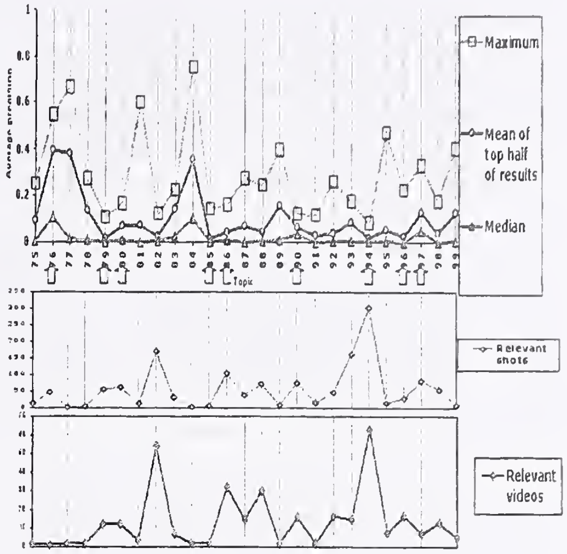


Figure 12: Manual search: average precision by topic



demonstrated. Similarly, the proper role and usefulness of the non-text topic elements is not yet clear. Matching the text of the topic against the text derived by automatic speech recognition on the video's audio track usually delivered better overall results than searches based on just the visual elements in the topic or combinations of the text and other elements. It is too early to draw convincing conclusions about either issue, but see the participants' papers for some interesting observations.

5 Approaches in brief

The following is a list of the groups that took part in one or more of the video track tasks and very short self-descriptions of the approaches taken by each participating research group. For detailed information the reader should consult the relevant system-specific paper in the proceedings.

5.1 Carnegie Mellon University (US)

The Informedia Project participated in the feature extraction task and both the manual and interactive search tasks. For the feature classification tasks, their standard approach was to hand label the feature training data using a labeling efficient interface, which allowed undergraduates to label one hour of video in 10 minutes for the presence/absence of one classification type (indoor, outdoor, etc.) They then extracted a set of standard low level image features such as HSV color histogram values, textures, EDH edge features, aggregrated line features, MPEG motion vectors and derived camera motion. These features were combined in a Support Vector Machine training process to produce a classification model for each category. Exceptions to this 'generic' image classifier approach were a custom developed face detector, a heuristic text detector and a decision-tree based people detector which used the face class as an input feature. Audio features were derived for the audio-based classes using a GMM model, and the monologue classifier combined both face output and audio features.

For the interactive track CMU used a modified version of the Informedia Digital Video Library System client, which was expanded to incorporate the classifier features and made more efficient to enable rapid display and exploration of large video data sets. It also incorporated an interface to multiple image search engines based on RGB or Munsell color, Texture, with different 3x3, 5x5, 7x7 blocks or QBIC-style image matching. An expert Informedia user, who did not have knowledge of the current TREC video collection, obtained the answers attempting to achieve high recall rather than speedy results. For the manual track, CMU submitted three systems: the first system was quite similar to last year's video track submission, combining speech recognition transcripts and OCR and image information in a linear fashion, while the second and best system extended the first system by incorporating the movie title and description information as text. This second system also added pseudo-relevance feedback for image retrieval as an additional combination module. Finally CMU submitted a third run using only the speech transcripts for text-only queries, without any relevance feedback or query expansion.

5.2 CLIPS-IMAG Grenoble (France)

This group used almost the same system for shot boundary detection as the one used for the TREC-2001 evaluation. This system detects "cut" transitions by direct image comparison after motion compensation and "dissolve" transitions by comparing the norms of the first and second temporal derivatives of the images. It also has a special module for detecting photographic flashes and filtering them as erroneous "cuts". Some parameters controlling the existing modules have been tuned using the TREC-2001 SBD corpus and reference segmentation, and a global parameter for the tuning of the recall versus precision compromise has been inserted.

The CLIPS group extracted only features 3 (faces), 4 (people), 8 (speech) and 10 (monologue). Face and people detection were based on a face detection tool publicly available from CMU run on one keyframe automatically extracted for each shot. The results were ranked according to the presence of a face and its size for feature 3 and according to the presence of at least two faces and the total size for feature 4. For features 8 and 10, they used the output of two different speech recognition systems, one from CLIPS-IMAG (GEOD team) and the other from LIMSI-CNRS, the same output as used by the group from Dublin. For feature 8, the length of detected speech segment within shots was used for ranking the results. For feature 10, the results were ranked using a combination of the length of a speech segment and the presence of a face.

Finally, CLIPS submitted three manual runs for the search task. One based only on speech transcription, on based only on a combination of donated features, and one based on a combination of both.

5.3 Dublin City University (Ireland)

DCU submitted results for three of the features from the feature set, namely speech, instrumental sound (music) and faces. Each technique worked directly on the encoded MPEG-1 bitstream. Speech extraction was based on measuring the duration of the rate of energy peaks of the audio signal. The same technique was extended to include rhythm and harmonicity for music detection while skin masks were used to detect the presence of faces. For the Search Task this group developed an interactive video retrieval system which used all 10 features identified earlier, three of which were the result of their own extractions, and the rest were donations from other groups. Twelve test users each ran the full 25 topics by formulating queries, browsing results and submitting results. The group ran two variations of their system, one which used the features plus the ASR transcript provided by LIMSI, and the other which used just the ASR transcript. All topic searches were limited to 4 minutes in total elapsed time.

5.4 Eurecom (France)

This group submitted runs under the feature extraction task. Their approach avoided complete decoding of the MPEG stream, basing decisions instead on the classification of the DCR macro-blocks — at some cost to the precision of the analysis. The work can be seen as an exploration of a "low-cost" baseline.

5.5 Fudan University (China)

Fudan University participated in the shot segmentation, feature extraction, and search tasks.

In the shot segmentation task, Fudan used most parts of their TREC-2001 shot segmentation system. The parameters used in the system were trained and adjusted based on the TREC-2001 video collection. According to the performance on TREC-2001 video collection, they selected the system parameters to generate the submissions. They added fade in/out detection to the system this year although the shot segmentation task did not include it. Evaluation showed that the system had a good balance between precision and recall. Comparing F-Value, the rank of the best result for all the changes, cut changes and gradual changes was 3, 3 and 9 (out of 54 systems). On gradual accuracy, frame-recall of the system was better than frame-precision. Compared with other submitted systems, their system was located at the middle in gradual accuracy.

In the feature extraction task, they developed a new video feature extraction system. It consisted of five sub-systems: outdoor / indoor detection, cityscape / landscape detection, face / people detection, text detection and speech / music / monologue detection. In each sub-system, a value calculated by whatever methods and features were used for ranking. Evaluation showed that the system worked well on these features: Cityscape, Landscape, Indoor and Music.

In the search task, Fudan submitted four runs. Considering the difficulty of search topics, they did not process all of the topics in each run. The whole architecture of the search system was almost the same as last year. However, there were some improvements in face recognition and object search. Fudan tried a fast manifold-based approach to face recognition in the TREC-2002 Search Task. This can be used when there are only few different images of a specific person and this process runs fast.

For each search topic, Fudan combined the similarities coming from different modules such as face recognition, text recognition, color histogram comparison, ASR text etc. In their submission, Sysl only used the information returned by their own search modules. There was no ASR Text and Feature Extraction results used. However, feature extraction confidence was useful for some topics. So in the runs labeled Sys2 and Sys3, they combined feature extraction confidence into the searching. Sys2 used their own feature extraction results and Sys3 used the reference feature extraction results provided by IBM and MediaMill. In Sys4, they combined the ASR Results provided by LIMSI. NIST's evaluation showed that their searching system was not effective in several topics. In their future work, Fudan plans to pay more attention to image similarity calculation.

5.6 IBM Research, Almaden and T.J. Watson (US)

IBM participated in the shot boundary detection, feature extraction and search tasks. This large group explored several diverse methods for video analysis, indexing, and retrieval, which included automatic descriptor extraction, statistical modeling, and multimodal fusion. In the shot boundary detection task, they explored several methods for making SBD more robust to poor video quality. Some of the methods explored include using localized edge gradient histograms and comparing pairs of frames at greater temporal distances. In the feature detection task the IBM group explored several methods for automatic descriptor extraction and statistical modeling and made significant efforts to manually annotate the Feature Training and Validation collections. First, using the Feature Training collection, they built statistical models of the concepts, exploring a variety of descriptors including color histograms, wavelet texture, edge histograms, color correlograms, motion vectors, audio spectrum features, and so on. They also investigated different discriminant modeling methods (e.g., support vector machines). Once the individual statistical models were constructed, they explored different fusion methods for maximizing retrieval effectiveness on the Feature Validation collection. The resulting fused classifiers were then applied to the Feature Test collection. Overall, feature detection results were submitted for all ten feature classes.

For the search task the IBM group investigated both manual and interactive methods of searching, submitting four runs as follows: (1) Manual searching using content-based retrieval (CBR) without knowledge of the Search Test collection; (2) Manual searching using spoken document retrieval (SDR) based on automatic speech recognition results; (3) A combination of CBR and SDR in manual searching; (4) Interactive use of CBR and SDR;

5.7 Imperial College London (United Kingdom)

Imperial College London used a shot-boundary detection scheme based on a multi-timescale detection algorithm in which colour histogram differences were examined over a range of frames. At each frame they calculated a distance measure for each of a range of timescales, and made decisions on whether a cut or gradual change had occurred according to where coincident peaks occurred in these distance measures. For the search task, they took a representative key frame for each shot and derived a number of low-level features including illumination-invariant colour representations, text from ASR and convolution filters. Query images were tested for similarity to a shot in the test set using the k-nearest neighbours approach. A novel relevance feedback system was then employed to allow the user to modify the query and update the results.

5.8 Indiana University (US)

At Indiana University researchers have developed a system named ViewFinder for the purpose of providing access to video content for a project named the Cultural digital Library Indexing Our Heritage (CLIOH). They took this existing system, made notable modifications, and applied it to the interactive search task, submitting one interactive search run.

5.9 Lowlands Group (the Netherlands)

This group participated in the search task by evaluating a probabilistic model for the retrieval of multimodal documents. The model was based on Bayes decision theory and combined models for text based search with models for visual search. The textual model, applied to the LIMSI transcripts, was based on the language modeling approach to text retrieval. The visual model, a mixture of Gaussian densities, described keyframes selected from shots. Both models had been proven successful on media specific retrieval tasks. Their contribution was the combination of both techniques in a unified model, ranking shots on ASR-data and visual features simultaneously. To further improve the query, they experimented with query expansion by adding additional example images found using Google image search. While the expansion process needed human involvement, they hoped the results would identify potential benefits of automatic expansion techniques for video search.

5.10 The MediaMill Group (the Netherlands)

The MediaMill Group performed feature extraction by evaluating a system aimed at training models for semantic concepts on a specific collection by active learning. The system was geared to feature classification for specific collections, to exploit characteristics of domain and collection, and to allow for user definition of problem-specific semantic concepts. Using the i-Notation system, annotators provided learning examples to the system in an efficient way. For active learning (i.e. classifier feedback during an annotation session) as well as final classification, a Maximum Entropy classifier was used. Binning was applied to provide the mapping of numerical values to binary values necessary for Maximum Entropy. A fixed pool of sixty visual descriptors was used as input for the Maximum Entropy classifier for all eight visual TREC features, so that extension of the approach to any other visual feature is trivial.

5.11 Microsoft Research Asia (China)

This team participated in the shot boundary, features and search tasks. For shot boundary detection, the submission was based on the last year's work but concentrated on improving gradual transition (GT) detection. The main feature for SBD was frame difference, the total difference of the binwise histogram comparison between two consequent frames in the R, G and B channels. Shot boundaries were then determined according to a set of heuristic rules. For feature extraction, multiple key frames were extracted for each shot and feature extraction was performed on these images. For the indoor, outdoor, cityscape and landscape features, trained models were employed based on color moment and edge direction histograms, aggregated over all keyframes from a shot. Face detection from keyframes and text overlay also ran on the multiple keyframes from each shot. The audio feature extraction was based on a support vector machine classifier with inputs based on low-level audio analysis.

This group used the Q-Video video retrieval system in the search task. Manual searching was performed using a combination of Color Moment (CM), Dominant Color (DC), HSV Histogram (HSVH), Color Layout (CL), Edge Histogram (EH), Color Texture Moment (CTM), Kirsh Direction Density (KDD), Wavelet feature (WF) and Motion Texture (MT) with different distance metrics employed for different feature sets. For interactive searching, users browsed retrieved shots and their feedback, both positive and negative, was fed into an SVM-based learning procedure for each topic, making it a kind of learning-based relevance feedback.

5.12 National University of Singapore (Singapore)

This group took part in the shot boundary detection task and used an expanded version of their previous temporal multi-resolution analysis (TMRA) work by introducing a new feature vector based on motion, incorporating functions to detect flash and camera/object motion, and selecting automatic thresholds for noise elimination based on the type of video. The framework can be used to extract meaningful keyframes and provides a unified approach to detection of gradual transitions and cuts.

5.13 Prous Science (Spain)

This company submitted runs under the search task but details have not been provided in writing at the time of writing this summary report. An overview paper describing the approach taken by Prous Science may become available along with other video track site reports, at a later time.

5.14 The University of Oulu (Finland)

The MediaTeam research group participated in collaboration with VTT Technical Research Centre of Finland to do the feature extraction, manual and interactive search tasks. In the feature extraction task they participated in detecting people, cityscapes, landscapes, speech and instrumental sound. The visual features used were based on spatio-temporal correlation of oriented gradient edge directions. Features from the audio signal consisted of various statistical measurements from signal power and energy. Representative shots for each feature class were selected from the feature development set to guide the vision-based feature detection. This group's video browsing and retrieval system contains a multi-modal indexing structure to access video shots. It uses combinations of self-organizing feature maps and semantic filters in content-based topic queries. It also provides a novel way to navigate interactively through vast collection of video shots based on a lattice-shaped browsing view. The view combines temporal coherence with metric shot similarities.

5.15 RMIT University (Australia)

RMIT participated in the shot boundary detection task, where they used the techniques of query by example (QBE) and ranked results, both often used in content-based image retrieval (CBIR). Each frame in turn was considered as an example query on the image collection formed by the other frames within a moving window. Transitions were detected by monitoring the relative ranks of these frames in the results list.

5.16 University of Bremen (Germany)

This group submitted runs under the shot boundary detection and feature detection tasks.

The shot detection approach was based on histogram differences. It was divided into two steps - feature extraction and shot boundary detection. Firstly, the histogram differences were calculated for the entire video in real time. Secondly, shot boundaries were detected. The advantage of this approach was the possibility to set adaptive thresholds for the shot boundary detection considering all extracted features of the complete video sequence. The adaptive threshold was set to a percentage of the maximum of all calculated difference values of the video. In the case of gradual changes, often multiple shot boundaries were detected. Therefore multiple detected shot boundaries that followed each other within a short temporal interval were grouped together and a gradual change was detected beginning with the first and ending with the last shot boundary in the interval.

For the feature extraction task the group examined whether it was possible to classify indoor and outdoor shots by their color distribution. In order to analyze the color distribution, first order statistical features were used, which were extracted from the histograms of the three color channels (RGB) and the grey level histogram. The features calculated from each histogram were average, variance, and amount of peaks, normalized to an interval [0...1]. In order to classify the shots into indoor and outdoor shots, a feed forward neural net with backpropagation learning was trained. At the input layer the 12 statistical features mentioned above were presented. The output layer consisted of two neurons that take on values between 0 and 1 measuring the probability for the features indoors or outdoors to be present in the shot. Two hidden layers each with 20 neurons were initialized with random weights. In order to train the neural net, some videos from the feature development collection were chosen. The shots were classified manually to generate 323 training data sets, 178 for indoors and 145 for outdoors.

In order to classify the shots from the feature extraction test collection, a set of n key frames was extracted from each shot. Every k-th frame of a shot was used as a key frame, but in order to be more independent of inaccuracies during the shot detection and of gradual changes (e.g., wipes, fades, or dissolves) a number of frames around the shot boundaries were skipped. In order to classify a shot, the set of n key frames was presented to the neural net. For each of the two output neurons a list was obtained containing n values, one for each key frame. The median for each list was calculated to obtain the final probabilities for the shot to be indoors or outdoors. In order to measure the accuracy of the classification result, the difference between the median values of the indoors and the outdoors neuron was calculated. If the difference exceeded a threshold the shot was classified to contain the feature with the higher probability. The difference was also used for the ranking.

5.17 University of Maryland (US)

The University of Maryland led a team made up of researchers from INSA Lyon (France) and the Universities of Maryland (US) and Oulu (Finland), and participated in the text feature extraction task and the search task. For search they provided a weighted query mechanism by integrating 1) text (OCR and ASR) content using full text and n-grams through the MG system, 2) color correlogram indexing of shots and images reported last year in TREC, and 3) ranked versions of the extracted binary features. All of the features are normalized, and a variety of distance measures are used to index into the collection. The command line version of the interface allowed users to make various queries, store them and use weighted combinations to generate a compound query.

In their interactive search experiments, most users generated their initial manual queries with the command line interface, and then explored a ranked collection of clips with an interactive interface. The interactive interface treated each video clip as a visual object in a multi-dimensional space, and each "feature" of that clip was mapped to one dimension. The user could visualize in two dimensions by placing any two features on the horizontal and vertical axis. Additional dimensions could be visualized by adding attributes to each object. Color, for example, could be used to represent a third feature dimension, size a fourth and shape a fifth dimension. Dynamic range sliders were provided for all features.

6 Summing up and moving on

This overview of the TREC-2002 Video Track has provided basic information on the track structure, data, evaluation mechanisms and metrics used, and a snapshot of what most of the participants did in their experiments. Further details about a particular group's approach and performance can be found in that group's site report. The raw results for each submitted run can be found in the results section of the final proceedings or under "Publications" on the trec.nist.gov website.

In 2003 the track will become an independent evaluation with a one- or two-day workshop (TRECVID 2003) immediately preceding TREC. The guidelines will be developed during the first quarter of 2003. The following are likely:

- using 120 hours of 1998 news video (MPEG-1) in 2003 and more of the same/similar in 2004
- continuing the three basic tasks: segmentation, feature extraction, search
- perhaps attempting detection of higher-level segments: stories, scenes
- keeping most of the features, but adding some appropriate to news
- striving for better system comparability in the search task
- creating more topics, perhaps 50, unbiased by detailed knowledge of the test collection
- significantly increasing the sizes of the search and especially the feature test collections.

The latest information about the TREC video retrieval evaluation efforts, past and present, is available from the track website at www-nlpir.nist.gov/projects/trecvid.

7 Authors' note

We are particularly grateful to Rick Prelinger and Niall O'Driscoll for their help with the Internet Archive data. We appreciate Jonathan Lasko's painstaking creation of the shot boundary truth data. The track would not have been possible without the software development work and general collaboration of Ramazan Taban, who has returned home to France and the job market. Our thanks to John Garofolo and Jose Joeman for their helpful suggestions on an earlier draft.

Finally, we would like to thank all the track participants and other contributors on the mailing list, and especially those groups who provided shot boundary and feature extraction output for use by others. These combined efforts made this running of the track possible. The spirit of the track was again a very positive one.

References

Aigrain, P., & Joly, P. (1994). The automatic real-time analysis of film editing and transition effects and its applications. *Computers and Graphics*, *18*(1), 93—103.

Armitage, L. H., & Enser, P. G. B. (1996). *Information Need in the Visual Document Domain: Report on Project RDD/G/235 to the British Library Research and Innovation Centre.* School of Information Management, University of Brighton.

Boreczky, J. S., & Rowe, L. A. (1996). Comparison of video shot boundary detection techniques. In I. K. Sethi & R. C. Jain (Eds.), *Storage and Retrieval for Still Image and Video Databases IV, Proc. SPIE 2670* (pp. 170–179). San Jose, California, USA.

Browne, P., Smeaton, A. F., Murphy, N., O'Connor, N., Marlow, S., & Berrut, C. (2000). Evaluating and Combining Digital Video Shot Boundary Detection Algorithms. In *IMVIP 2000 - Irish Machine Vision and Image Processing Conference.* Belfast, Northern Ireland: URL: www.cdvp.dcu.ie/Papers/IMVIP2000.pdf.

Enser, P. G. B., & Sandom, C. J. (2002). Retrieval of Archival Moving Imagery — CBIR Outside the Frame. In M. S. Lew, N. Sebe, & J. P. Eakins (Eds.), *Image and Video Retrieval, International Conference, CIVR 2002, London, UK, July 18-19, 2002, Proceedings* (Vol. 2383). Springer.

Ford, R. M. (1999). A Quantitative Comparison of Shot Boundary Detection Metrics. In M. M. Yueng, B.-L. Yeo, & C. A. Bouman (Eds.), *Storage and Retrieval for Image and Video Databases VII, Proceedings of SPIE Vol. 3656* (pp. 666–676). San Jose, California, USA.

The Internet Archive Movie Archive home page. (2002). URL: http://www.archive.org/movies/.

Lee, A. (2001). *VirtualDub home page.* URL: www.virtualdub.org/index.

Marchionini, G. (2001). *The Open Video Project home page.* URL: www.open-video.org.

Ruiloba, R., Joly, P., Marchand-Maillet, S., & Quénot, G. (1999). Towards a Standard Protocol for the Evaluation of Video-to-Shots Segmentation Algorithms. In *European Workshop on Content Based Multimedia Indexing.* Toulouse, France: URL: clips.image.fr/mrim/georges.quenot/articles/cbmi99b.ps.

Shatford, S. (1986). Analyzing the Subject of a Picture: A Theoretical Approach. *Cataloging and Classification Quarterly, 6*(3), 39—61.

Smeaton, A., Over, P., & Taban, R. (2002). The trec-2001 video track report. In E. M. Voorhees & D. K. Harman (Eds.), *The Tenth Text REtrieval Conference (TREC-2001).* Gaithersburg, MD, USA.

Overview of the TREC-2002 Web Track

Nick Craswell and David Hawking
CSIRO Mathematical and Information Sciences,
Canberra, Australia
{Nick.Craswell,David.Hawking}@csiro.au

April 16, 2003

Abstract

The TREC-2002 Web Track moved away from non-Web relevance ranking and towards Web-specific tasks on a 1.25 million page crawl ".GOV". The topic distillation task involved finding pages which were relevant, but also had characteristics which would make them desirable inclusions in a distilled list of key pages. The named page task is a variant of last year's homepage finding task. The task is to find a particular page, but in this year's task the page need not be a home page.

1 Introduction

The TREC-2002 Web Track activities centred on two tasks: A Topic Distillation Task and a Named Page Finding Task. Both made use of an 18 gigabyte, 1.25 million document 2002 partial crawl of the .gov domain, distributed on CD-ROM as the .GOV collection.

2 Guidelines

2.1 This Year's Aims

1. To begin work with a new (early 2002) crawl of an important Web domain (.gov). Past TREC Web experiments used data from 1997.
2. To formulate Web-specific search tasks, which are representative of common Web search activities, leading to new evaluation methods and new effective Web retrieval algorithms.
3. To conduct topic distillation experiments, in order to understand the selectivity required to generate a short top-N list, even when a very large set of on-topic documents are available.
4. To conduct named page experiments, to find if there are particular forms of ranking evidence which help us to find specific Web documents (last year's experiments found that URL type and anchor text were useful for finding homepage documents).
5. To make available the first set of reusable relevance judgments for the new .GOV test collection.

2.2 Dataset

The .GOV corpus is a crawl of Web sites in the .gov domain from early 2002. That makes it 5 years newer than previous TREC Web collections, all of which were based on a 1997 Internet Archive crawl. Although we hope that the most useful of the Web search techniques would work on 1997 crawls as well as 2002, it is also highly desirable to have a dataset representative of the current Web. Some properties of .GOV are listed in Table 1.

Table 1: Salient properties of the .GOV corpus. (Mime types as reported by the servers.)

Number of pages	1,247,753
Number of pages by mime type:	
text/html	1,053,110
application/pdf	131,333
text/plain	43,753
application/msword	13,842
application/postscript	5,673
other (containing text)	42
Average page size	15.2 kB
Number of hostnames	7,794
Total number of links	11,164,829
Number of cross-host links	2,470,109
Average cross-host links per host	317

The crawl included binary and text mime types, and was stopped after 1 million HTML pages. The HTML and text, plus extracted text of other document types, gives a total of 1.25 million documents. Total data size was 35 gigabytes, which we considered too great an increase in corpus size (over WT10g), so a 100 kilobyte cutoff was applied to all documents, reducing the total size to 18 gigabytes.

The .GOV dataset is distributed by CSIRO [3]. Note that the standard distribution includes the HTML documents plus text extracted from other formats such as PDF. The original PDFs and other binary files such as images were collected and are potentially available. However, the full crawl including binaries is 67 gigabytes almost four times the size of the collection as distributed (and it would not compress as well).

Docids are 14 characters and of the form G09-04-2395783, meaning that this document is in the bundle G00/04.gz at byte offset 2395783. All .GOV documents can be located in this way via their docid. The collection was distributed on seven CDs, with an eighth containing tables of URLtoID, links, duplicates and redirects. The URLtoID table lists all valid .GOV docids and their corresponding URLs. The link table is useful for link-based ranking experiments, and a potentially more complete picture of link structure could be built in conjunction with the provided duplicate and redirect tables. These give information on link target URLs visited by the crawler but not included in .GOV because they contained duplicate content or forwarded users to another URL.

The .GOV corpus has fewer documents than WT10g, but has a much larger average document size (15k vs 7k), reflecting changes in Web authoring over the space of five years (perhaps the prevalence of navigation bars and scripting in more recent pages). Compared to WT10g, .GOV also has the strictest file-type checking of any Web collection so far, leading to very few binaries in the corpus.

We chose to crawl .gov for several reasons. It is a commercially interesting domain, meaning that important services are provided based on precisely this sort of crawl. It is also a crawl of manageable size, in that it can be distributed on CD and is within the data size limitations of most TREC systems. By contrast the crawl of a large search engine would be perhaps 30 terabytes, well beyond the bounds of manageability using current technologies (the 100 gigabyte VLC2 is still considered large relative to the storage media and systems available to researchers). Luckily, many smaller crawls such as .GOV are conceivable, which are of manageable size and of significant research and commercial interest. The .GOV crawl is also of a size which allows sufficiently complete relevance judgments.

2.3 Topic Distillation Task

Topics: 551-600 Example:

```
<top>
```

```
<num> Number: 600
<title> highway safety
<desc> Description:
Find documents related to improving highway safety in the U.S.
<narr> Narrative:
Relevant documents include those related to the improvement of safety
of all vehicles driven on highways, including cars, trucks, vans, and
tractor trailers. Ways to reduce accidents through legislation,
vehicle checks, and drivers education programs are all relevant.
</top>
```

The premise of the topic distillation task is that some quality, in addition to relevance, is desirable in Web search results lists. This quality has been called authority, quality, definitiveness and many other names in previous studies [4, 1, 2]. Assuming it exists, systems will have to find evidence which indicates its presence, and strike a balance between relevance and "quality" in search algorithms. This balancing act is analogous to the balance between newness and relevance when searching a news archive: it is desirable to return documents which are both relevant and new (if any).

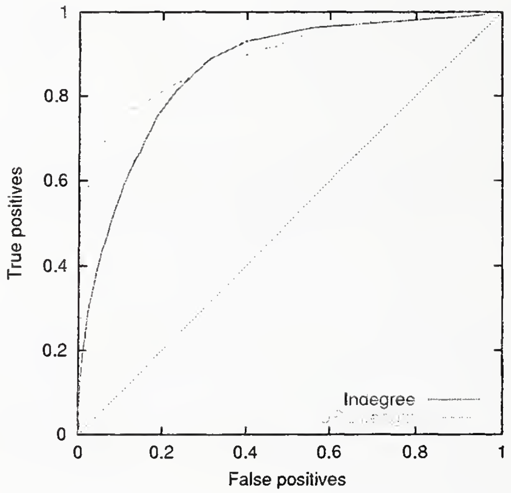


Figure 1: Query-independent properties of .GOV pages predict which will be listed in Web directories.

In our non-TREC research, we have found some evidence that query-independent evidence can indicate desirability, based on analysis of hand-made URL lists. We sorted all .GOV URLs in reverse URL length order and link indegree order. Then we found examples of "quality URLs" in .GOV, by identifying those which are hand-listed in the Yahoo! and DMOZ Web directories. Web directories are an alternative Web search technology, where within a category hierarchy, each category has a list of URLs created by a human editor. Figure 1 shows that the URL and link orderings were good predictors of hand-listing. If they predict hand-listing, then these characteristics might also be good predictors of which search results would be preferred by Web search users (who, after all, are also the audience of the Web directories).

In the topic distillation task we evaluate systems in terms of their ability to return relevant "key pages". A key page is one which the relevance assessor would find worthy of including in a short list of important URLs (the sort of choice made by Web directory editors). The "relevant key pages" found by assessors should thus be relevant and posses that special quality which makes pages worthy of inclusion in a short list. We did not go further than this in defining what makes a page list-worthy, since it has not been agreed in the research community what the definition is (quality, authority, definitiveness etc), and we did not want to bias assessments. The main measure is precision at 10.

2.4 Named Page Finding Task

Topics: NP1-NP150 Example:

```
<top>
<num> Number: NP1
<desc> Description:
America's Century Farms
</top>
```

The objective in the named page finding task is to find a particular Web page in .GOV, given a query which describes it by name. For example, the query "America's Century Farms" might lead to a particular .GOV Web page describing farms that have remained in one family's hands for over 100 years. The assessment task was simply to identify any duplicate URLs for the named page, since a page can appear at more than one URL. The main measure was the reciprocal rank of the first correct answer.

2.5 Indexing Restrictions

There were none. Participants were permitted to index all of each document or exclude certain fields as they wished.

2.6 Submissions and Judgments

Runs were received from a total of 23 groups: ajou, chinese_academy, city-pliers, cmu_lti, csiro, cuny, dgic_stokoe, fudan, glasgow, hummingbird, ibm-haifa, iit, illinois_chicago, irit, kasetsart, lit_singapore, neuchatel, tsinghua, umbc-cost, umelbourne, uva, waterloo, yonsei.

2.7 Topic Distillation Task

Seventy-one official runs were submitted from seventeen participating groups. The number of pages judged was 56,650 of which 1574 were judged to meet the criteria. Figure 2 shows the distribution of numbers of key resources found per topic. For a few topics the number of such resources is very much higher than expected.

While pages hand-listed in Web directories tended to have short URLs and high indegree (Figure 1), key resources from this year's track did not show such tendencies as strongly (Figure 3).

2.8 Named Page Finding Task

Seventy official runs were submitted from eighteen participating groups.

Only one correct answer was identified for most of the 150 topics, but there were three correct answers to two topics (9 and 145) and two for 16 topics (1, 8, 14, 24, 26, 50, 51, 63, 66, 67, 68, 85, 89, 128, 138, and 146).

3 Results

3.1 Topic Distillation Task

Full official results for the Topic Distillation task are reported in Appendix 1. Results on a per-group basis are presented in Table 2. Here we briefly summarize the information available about the experiments conducted by the top five groups

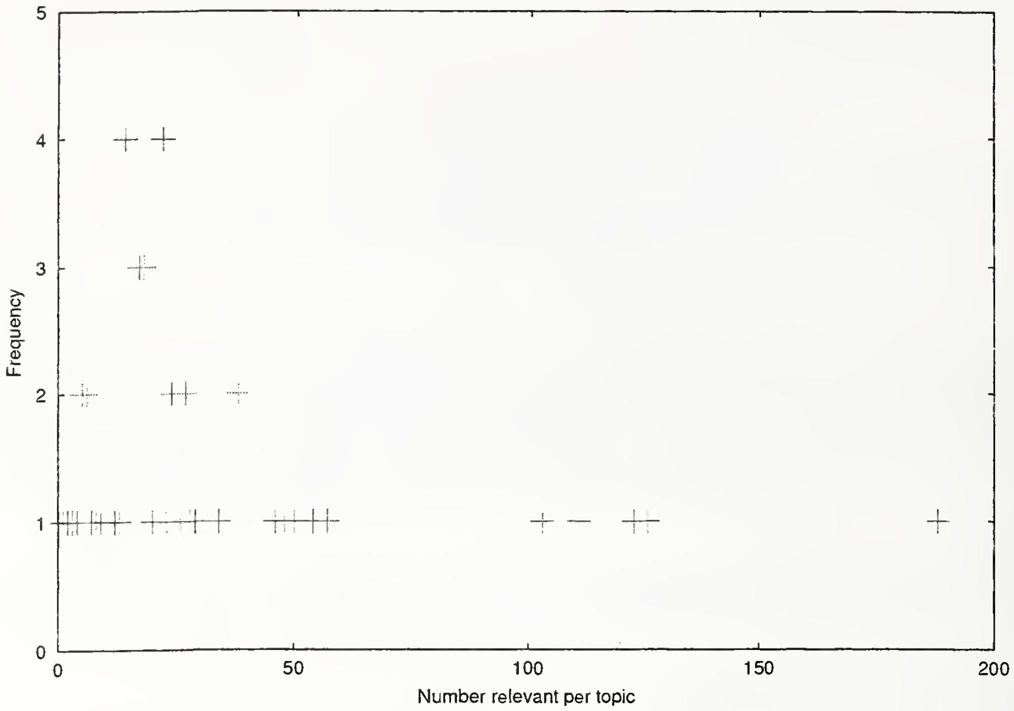


Figure 2: Topic distillation task: Number of key resources per topic.

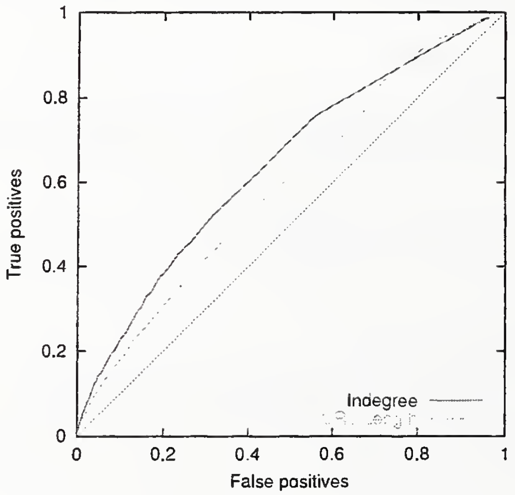


Figure 3: Query-independent properties which were good predictors in Figure 1 are less useful when predicting this year's key pages.

TsingHua University TsingHua used Okapi with stemming and Fox stoplist but no query expansion or feedback.

They explored:

1. techniques based on link structure and link text, especially the use of out-degree to find key resources;

Table 2: P@10 results for the best topic distillation run submitted by each participating group. The codes D, A, L indicate the use of document structure (D), Anchor text (A) and Link structure (L).

Rank	P@10	Group	Best Run	Run type	D?	A?	L?	#Runs
1.	0.2510	tsinghua	thutd5	realistic	D	A	_	5
2.	0.2408	city-pliers	pltr02wt2	realistic	_	_	_	2
3.	0.2306	chinese_academy	icttd1	realistic	_	_	_	2
4.	0.2286	ibm-haifa	ibmhaifapr	realistic	D	A	L	4
5.	0.2224	glasgow	uog05tad	realistic	D	A	L	2
6.	0.2163	irit	mercah	realistic	_	_	_	2
7.	0.1959	neuchatel	uninedi5	realistic	D	A	_	5
8.	0.1939	fudan	fduwt11t1	realistic	D	_	L	3
9.	0.1939	umelbourne	mu525	realistic	_	_	_	5
10.	0.1755	uva	uamst02wtt	realistic	_	_	_	5
11.	0.1510	yonsei	yedi01	realistic	D	A	L	1
12.	0.1143	umbc-cost	carrot2a	realistic	_	_	_	1
13.	0.1082	cuny	pirc2wd2	realistic	D	A	L	2
14.	0.1041	illinois_chicago	uic0104	realistic	_	_	L	2
15.	0.1000	csiro	csiro02td1	realistic	_	_	L	1
16.	0.0714	dgic_stokoe	tdwsdtfidf	realistic	_	_	_	2
17.	0.0571	ajou	ajouai0210	realistic	_	_	L	4

2. the roles of different HTML fields in ranking content;
3. post-processing of retrieval results, namely a site uniting approach
4. A genetic algorithm based dynamic parameter learning approach.

They found that anchor text was useful but out-degree was not. They also found that site uniting methods which worked well on the small number of training examples improved average precision but not P@10. Parameter settings learned on past Web tasks did not improve performance this year.

City University, London It is significant that the second best performing run (`pltr02wt2` from City University, London) was a straightforward content retrieval run based on Okapi BM25 (with non-default parameter for parameter b and stemming but no relevance feedback.

Chinese Academy No details available.

IBM Haifa Query expansion via lexical affinities. Knowledge Agents and Knowledge Bases incorporating content scores, anchor text, Kleinberg Hub and Authority scores and SALSA scores. Site compression. Title filtering - eliminate documents which have no query word in their title (beneficial). Duplicate elimination based on textual similarity (harmful).

Glasgow University Experimentation focused on:

1. A probabilistic framework for combining link and content, called the Absorbing Model, based on Markov chains and applicable in either static or dynamic forms;
2. A spreading activation method (either query independent or query dependent) for detecting site entry points;
3. Anchor text.
4. A genetic algorithm based dynamic parameter learning approach.

They found that body only indexing and link analysis without anchors work well while spreading activation on sites was equivocal and query expansion and PageRank were detrimental.

IBM Haifa identified a scoring problem in that no penalty was applied to runs which included multiple duplicates or near duplicates.

3.2 Named Page Finding Task

Table 3: MRR results for the best named page finding run submitted by each participating group. The codes D, A, L indicate the use of document structure (D), Anchor text (A) and Link structure (L).

Rank	MRR	Group	Run	Run Type	D?	A?	L?
1	0.719	tsinghua	thunp3	realistic	D	A	-
2	0.676	cmu_lti	lmralleq	realistic	D	A	-
3	0.671	yonsei	yenp01	realistic	D	A	L
4	0.654	glasgow	uog07cta	realistic	D	A	-
5	0.636	neuchatel	uninenp1	realistic	D	A	-
6	0.626	hummingbird	hum02pd	realistic	D	-	-
7	0.613	chinese_academy	ictnp6	realistic	D	A	-
8	0.587	iit	iit02b	realistic	-	-	-
9	0.578	lit_singapore	litlink	realistic	D	A	L
10	0.576	umelbourne	mu106	realistic	D	A	-
11	0.573	csiro	csiro02np01	realistic	-	-	-
12	0.564	illinois_chicago	uicnp03	realistic	-	-	-
13	0.535	waterloo	uwmtbw2	realistic	-	A	L
14	0.432	uva	uamst02wntla	realistic	-	A	-
15	0.418	city-pliers	pltr02wt9	realistic	-	-	-
16	0.263	cuny	pirc2wnp1	realistic	D	A	-
17	0.132	ajou	ajouai0204	realistic	D	-	-
18	0.010	kasetsart	kuhpf0201	realistic	-	A	-

Full official results for the Named Page Finding task are reported in Appendix 2. Results on a per-group basis are presented in Table 3. Here we briefly summarize the information available about the experiments conducted by the top five groups

TsingHua University They built a collection of surrogate documents comprising keywords, titles and incoming anchor text. Ranks obtained with these surrogates were combined with ranks from the original documents, using $S' = a*1/rank1+(1-a)*1/rank2$. (Note that the original collection was divided into two sub-collections: html and non-html and the results merged using a novel procedure (see the TsingHua paper for details). The combined score outperformed the original score which in turn outperformed the surrogate score.

CMU LTI Their basic model was a generative language model, where the language model for the document was a linear interpolation of several language models (title, in-link text, full text, meta tag text, image alt text, url text, large fonts). Using document structure in this way did improve performance over just using a simple language model.

They were unable to find useful prior probabilities for this task, either on training data created locally or on the test data. They tried in-link count, document length, document file type, and url length.

Yonsei No details available.

Glasgow University Anchor text proved to be more useful than link analysis, significantly improving results.

They found that body only indexing and link analysis without anchors work well while spreading activation on sites was equivocal and query expansion and PageRank were detrimental.

U. Neuchatel A second representation of each document in the .GOV collection was created, comprising the documents title and all its incoming anchor text. Okapi scores were computed for both representations and linearly combined $\alpha S_{content} + (1 - \alpha)S_{anchortitle}$ (without normalisation). The best results were obtained with $\alpha = 0.6$.

4 Conclusions

The .GOV corpus provided an interesting and realistic dataset for the purposes of the track. No significant problems were reported in working with it.

The Named Page Finding task was an interesting variant on earlier Home Page Finding evaluations. Unsurprisingly, URL-type analyses did not bring improvement in performance. However, several leading participants reported an improvement in performance by adding anchor text and structural information to a content-only run. In 2003, it is anticipated that a mixed Home Page / Named Page task might prove interesting.

The Topic Distillation task proved difficult to explain to both participants and assessors and there was considerable disparity between the interpretations of these two groups. It is not clear what, if any, conclusions can be drawn at this stage. The task is worth repeating in 2003 but more explanatory effort is needed.

Acknowledgements

We are very grateful to Charlie Clarke of the University of Waterloo for providing the crawler software used to collect the .GOV data and for supervising the crawl. We are also grateful to Ed Fox of Virginia Tech for providing the machine and network connection to support the crawl and to Ian Soboroff of NIST for facilitating the crawl and also for his invaluable assistance in organising the track.

References

[1] K. Bharat and M. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of ACM SIGIR'98*, pages 104–111, 1998.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of WWW7*, pages 107–117, 1998. `http://www7.scu.edu.au/programme/fullpapers/1921/com1921.htm`.

[3] CSIRO. TREC Web Tracks home page. `www.ted.cmis.csiro.au/TRECWeb/`.

[4] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

Appendix 1 - Topic Distillation runs

P@10 results for all topic distillation run submitted. The codes D, A, L indicate the use of document structure (D), Anchor text (A) and Link structure (L)

Rank	P@10	Group	Run	Run type	D?	A?	L?	#Runs
1.	0.2510	tsinghua	thutd5	realistic	D	A	-	5
2.	0.2408	city-pliers	pltr02wt2	realistic	-	-	-	2
3.	0.2306	tsinghua	thutd3	realistic	D	A	-	3
4.	0.2306	chinese_academy	icttd1	realistic	-	-	-	2
5.	0.2286	ibm-haifa	ibmhaifapr	realistic	D	A	L	4
6.	0.2245	tsinghua	thutd2	realistic	D	A	-	2
7.	0.2224	glasgow	uog05tad	realistic	D	A	L	2
8.	0.2163	irit	mercah	realistic	-	-	-	2
9.	0.2143	tsinghua	thutd4	realistic	D	A	-	4
10.	0.2122	ibm-haifa	ibmhaifat10	realistic	D	A	L	3
11.	0.2082	glasgow	uog04cta2dqh	realistic	D	A	L	4
12.	0.2061	ibm-haifa	ibmhaifat10d	realistic	D	A	L	3
13.	0.2000	city-pliers	pltr02wt1	realistic	-	-	-	1
14.	0.2000	city-pliers	pltr02wt4	realistic	-	-	-	4
15.	0.1980	tsinghua	thutd1	realistic	D	A	L	1
16.	0.1959	neuchatel	uninedi5	realistic	D	A	-	5
17.	0.1939	ibm-haifa	ibmhaifaap	realistic	D	A	L	2
18.	0.1939	fudan	fduwt11t1	realistic	D	-	L	3
19.	0.1939	fudan	fduwt11o1	realistic	D	-	L	1
20.	0.1939	glasgow	uog03ctadqh	realistic	D	A	L	1
21.	0.1939	umelbourne	mu525	realistic	-	-	-	5
22.	0.1939	fudan	fduwt11t2	realistic	D	A	L	2
23.	0.1898	ibm-haifa	ibmhaifabase	realistic	D	A	L	1
24.	0.1857	umelbourne	mu111	realistic	D	A	-	1
25.	0.1755	city-pliers	pltr02wt3	realistic	-	-	-	3
26.	0.1755	uva	uamst02wtt	realistic	-	-	-	5
27.	0.1755	chinese_academy	icttd2	realistic	-	-	-	1
28	0.1714	fudan	fduwt11o2	realistic	D	A	L	4
29.	0.1694	umelbourne	mu624	realistic	-	-	-	2
30.	0.1673	chinese_academy	icttd3	realistic	-	-	-	3
31.	0.1510	fudan	fduwt11b0	realistic	-	-	-	5
32.	0.1510	yonsei	yedi01	realistic	D	A	L	1
33.	0.1469	yonsei	yedi01no	realistic	D	A	L	2
34.	0.1429	irit	mercure	realistic	-	-	L	3
35.	0.1429	neuchatel	uninedi4	realistic	D	A	-	4
36.	0.1306	glasgow	uog01ctaiallh	realistic	D	A	L	5
37.	0.1163	umelbourne	mu313	realistic	D	A	-	3
38.	0.1143	umbc-cost	carrot2a	realistic	-	-	-	1
39.	0.1143	glasgow	uog02ctadh	realistic	D	A	L	3
40.	0.1082	umelbourne	mu212	realistic	D	A	-	1
41.	0.1082	irit	mercurelynx	realistic	-	-	L	1
42.	0.1082	cuny	pirc2wd2	realistic	D	A	L	2
43.	0.1041	illinois_chicago	uic0104	realistic	-	-	L	2
44.	0.1000	csiro	csiro02td1	realistic	-	-	L	1
45.	0.1000	illinois_chicago	uic0101	realistic	-	-	L	4
46.	0.1000	uva	uamst02wta	realistic	-	A	-	4
47.	0.0939	csiro	csiro02td5	realistic	-	-	L	5
48.	0.0898	illinois_chicago	uic0103	realistic	-	-	L	3
49.	0.0878	city-pliers	pltr02wt5	exploratory	-	-	-	5
50.	0.0837	neuchatel	uninedi1	realistic	D	A	L	1
51.	0.0816	cuny	pirc2wd1	realistic	D	A	L	1
52.	0.0796	illinois_chicago	uic0102	realistic	-	-	L	1
53.	0.0776	neuchatel	uninedi3	exploratory	D	A	L	2
54.	0.0714	csiro	csiro02td2	realistic	-	A	L	2
55.	0.0714	dgic_stokoe	tdwsdtfidf	realistic	-	-	-	2
56.	0.0714	umbc-cost	carrot2c	realistic	-	-	L	4
57.	0.0673	uva	uamst02wttri	realistic	-	-	L	3
58.	0.0653	uva	uamst02wtacs	realistic	-	A	-	2
59.	0.0653	dgic_stokoe	tdtfidf	realistic	-	-	-	1
60.	0.0633	uva	uamst02wtari	realistic	-	A	L	1
61	0.0571	ajou	ajouai0210	realistic	-	-	L	4
62.	0.0551	umbc-cost	carrot2d	realistic	-	-	L	2
63.	0.0551	ajou	ajouai0206	realistic	-	-	L	2
64.	0.0551	umbc-cost	carrot2b	realistic	-	-	-	3
65.	0.0531	ajou	ajouai0207	realistic	-	-	L	1
66.	0.0347	ajou	ajouai0208	realistic	-	-	L	3
67.	0.0327	neuchatel	uninedi2	realistic	D	A	L	3
68.	0.0245	ajou	ajouai0209	realistic	-	-	L	5
69.	0.0184	csiro	csiro02td4	realistic	-	A	L	4
70.	0.0184	umbc-cost	carrot2e	realistic	-	-	L	4
71.	0.0184	csiro	csiro02td3	realistic	-	-	L	3

Appendix 2 - Named Page Finding runs

MRR results for all named page finding run submitted. The codes D, A, L indicate the use of document structure (D), Anchor text (A) and Link structure (L).

Rank	MRR	Group	Run	Run Type	D?	A?	L?
1	0.719	tsinghua	thunp3	realistic	D	A	-
2	0.717	tsinghua	thunp5	realistic	D	A	-
3	0.690	tsinghua	thunp1	realistic	D	A	-
4	0.687	tsinghua	thunp4	realistic	D	A	-
5	0.676	cmu_lti	lmralieq	realistic	D	A	-
6	0.671	yonsei	yenp01	realistic	D	A	L
7	0.667	cmu_lti	lmrallest	realistic	D	A	-
8	0.654	glasgow	uog07cta	realistic	D	A	-
9	0.651	glasgow	uog10ctad	realistic	D	A	L
10	0.643	glasgow	uog09cta2	realistic	D	A	-
11	0.636	neuchatel	uninenp1	realistic	D	A	-
12	0.626	hummingbird	hum02pd	realistic	D	-	-
13	0.625	neuchatel	uninenp3	realistic	D	A	-
14	0.616	neuchatel	uninenp2	realistic	D	A	-
15	0.613	chinese_academy	ictnp7	realistic	D	A	-
16	0.613	chinese_academy	ictnp6	realistic	D	A	-
17	0.611	cmu_lti	lmrnostruct	realistic	-	A	-
18	0.589	cmu_lti	lmrsmall	realistic	D	A	-
19	0.587	iit	iit02b	realistic	-	-	-
20	0.580	iit	iit02tfa	realistic	D	A	-
21	0.578	lit_singapore	litlink	realistic	D	A	L
22	0.576	umelbourne	mu106	realistic	D	A	-
23	0.576	iit	iit02tf	realistic	D	-	-
24	0.573	csiro	csiro02np01	realistic	-	-	-
25	0.568	cmu_lti	lmrdocstruct	realistic	D	-	-
26	0.564	illinois_chicago	uicnp03	realistic	-	-	-
27	0.559	chinese_academy	ictnp2	realistic	D	A	-
28	0.557	chinese_academy	ictnp3	realistic	D	A	-
29	0.555	chinese_academy	ictnp4	realistic	D	A	-
30	0.552	glasgow	uog06c	realistic	-	-	-
31	0.550	illinois_chicago	uicnp02	realistic	-	-	-
32	0.538	hummingbird	hum02upd	realistic	D	-	-
33	0.535	waterloo	uwmtbw2	realistic	-	A	L
34	0.530	tsinghua	thunp2	realistic	D	A	-
35	0.527	hummingbird	hum02up	realistic	D	-	-
36	0.524	umelbourne	mu609	realistic	D	A	-
37	0.524	umelbourne	mu208	realistic	D	A	-
38	0.516	glasgow	uog08ctap	realistic	D	A	-
39	0.509	waterloo	uwmtbw0	realistic	-	-	-
40	0.504	neuchatel	uninenp4	realistic	D	A	-
41	0.495	illinois_chicago	uicnp01	realistic	-	-	L
42	0.456	hummingbird	hum02ud	realistic	-	-	-
43	0.432	uva	uamst02wntla	realistic	-	A	-
44	0.427	lit_singapore	littext	realistic	-	-	-
45	0.425	uva	uamst02wntl	realistic	-	-	-
46	0.418	city-pliers	pltr02wt9	realistic	-	-	-
47	0.416	csiro	csiro02np03	realistic	D	-	-
48	0.416	city-pliers	pltr02wt8	realistic	-	-	-
49	0.414	city-pliers	pltr02wt7	realistic	-	-	-
50	0.402	umelbourne	mu80a	realistic	-	-	-
51	0.367	uva	uamst02wntma	realistic	-	A	-
52	0.337	hummingbird	hum02uhp	realistic	D	-	-
53	0.334	city-pliers	pltr02wt6	realistic	-	-	-
54	0.328	uva	uamst02wna	realistic	-	A	-
55	0.318	csiro	csiro02np04	realistic	D	A	-
56	0.307	csiro	csiro02np16	realistic	D	A	L
57	0.263	cuny	pirc2wnp1	realistic	D	A	-
58	0.260	uva	uamst02wntm	realistic	-	-	-
59	0.241	csiro	csiro02np02	realistic	-	A	-
60	0.207	umelbourne	mu307	realistic	D	A	-
61	0.150	waterloo	uwmtbw1	realistic	-	-	L
62	0.132	ajou	ajouai0204	realistic	D	-	-
63	0.108	ajou	ajouai0201	realistic	D	-	-
64	0.106	waterloo	uwmtbw4	realistic	-	A	L
65	0.103	waterloo	uwmtbw3	realistic	-	A	L
66	0.076	cuny	pirc2wnp2	realistic	D	A	L
67	0.072	ajou	ajouai0202	realistic	D	-	-
68	0.071	ajou	ajouai0203	realistic	D	-	-
69	0.010	kasetsart	kuhpf0201	realistic	-	A	-
70	0.010	ajou	ajouai0205	realistic	D	-	L

TREC2002 QA at BBN:
Answer Selection and Confidence Estimation

Jinxi Xu, Ana Licuanan, Jonathan May, Scott Miller and Ralph Weischedel
BBN Technologies
50 Moulton Street
Cambridge, MA 02138

Abstract

We focused on two issues: answer selection and confidence estimation. We found that some simple constraints on the candidate answers can improve a pure IR-based technique for answer selection. We also found that a few simple features derived from the question-answer pairs can be used for effective confidence estimation. Our results also confirmed findings by Dumais et al, 2002 that the World-Wide Web is a very useful resource for answering TREC-style factoid questions.

1. Introduction

Answer selection and confidence estimation are two central issues in question answering (QA). The goal of answer selection is to choose from a pool of answer candidates the most likely answer for a question. The problem of confidence estimation is to compute $P(correct|Q, A)$, the probability of answer correctness given a question Q and an answer A. Showing an incorrect answer has negative impact because it not only burdens and but also may mislead the user with incorrect information. Confidence estimation is important because a QA system relies on it to decide whether or not to show the user an answer.

For answer selection, we used a HMM-based IR system (Miller et al, 1999) to first select documents that are likely to contain answers to a question and then rank candidate answers based on the answer contexts using the same IR system. Then we used a few constraints to re-rank the candidates. Such constraints include whether a numerical answer quantifies the correct noun, whether the answer is of the correct location sub-type and whether the answer satisfies the verb arguments of the question.

For confidence estimation, direct estimation of $P(correct|Q,A)$ is impossible because it would require virtually unlimited training data. Instead, we computed the probability based on a few features that concern Q and A. The features were empirically selected with two criteria in mind: being able to predict answer correctness and having a small dimensionality. The features include the type of the question, the number of matched question words in the answer context and whether the answer satisfies the verb arguments of the question.

We also experimented with using the World Wide Web to supplement the TREC corpus for QA. Our results confirmed the positive findings reported in earlier studies (Dumais et al, 2002). We also found that the frequency of an answer in the returned Web pages is a strong predictor of answer correctness.

We submitted three runs: BBN2002A, BBN2002B and BBN2002C. BBN2002A is our base run, which did not use the Web for answer finding and confidence estimation. Both

BBN2002B and BBN2002C used the Web, but they used slightly different methods for confidence estimation. In our experiments, we used the TREC9&10 questions for estimating the parameters that were used in confidence estimation. To be consistent with the TREC 2002 QA track, only factoid questions in TREC9&10 were used for parameter training.

2. The Base Run: BBN2002A

2.1 Answer Selection

Selecting the best answer for a question from the TREC corpus takes the following steps. First we used BBN's IR system (Miller et al, 1999) to select the top *n* documents from the TREC corpus. For the training questions, we set *n*=100, for efficiency considerations. For the test questions (i.e. TREC 2002 questions), we set *n*=300.

The question was then typed. A question classifier automatically assigned the question to one of the 30 types defined in our answer type taxonomy. (In some rare cases a question was assigned to more than one type. For convenience of discussion, we will assume one type per question). Similar to taxonomies used in other QA systems, ours includes common named entities such as persons, dates, locations, numbers, monetary amounts and so forth.

Then the candidate answers were ranked. The pool of candidates consists of occurrences of named entities in the top documents that match the answer type of the question. Named entities in the documents were recognized using BBN's IdentiFinder system (Bikel et al, 1999). The candidates were first ranked using BBN's IR system. To score a candidate, every text window that has the candidate at the center and has fewer than 60 words (for efficiency considerations) was scored against the question by the IR engine. The score for the candidate took that of the highest-scored window. The purpose of using multiple passages is to avoid choosing the optimal passage length, which is known to a tricky problem. A similar strategy was used by Kaszkiel for document retrieval (Kaszkiel & Zobel, 2001).

Then the candidates were re-ranked, by applying the following constraints:

1. If the question asks for a number, check whether the answer quantifies the same noun in the answer context as in the question.
2. If the question looks for a sub-type of locations (e.g. a country, state or city), check whether the answer is of that sub-type. We employed lists of countries, states and cities for this purpose. This constraint is useful because our taxonomy does not distinguish different kinds of locations.
3. Check if the answer satisfies the verb arguments of the question. For example, if the question is "Who killed X", a preferred candidate should be the subject of the verb "killed" and X should be the object of "killed" in the answer context. Verb arguments were extracted from parse trees of the question and the sentences in the corpus. We used BBN's SIFT parser (Miller et al, 2000) for verb argument extraction.

Candidates that satisfy the above constraints were ranked before those that do not. The highest ranked candidate was chosen as the answer for the question. The constraints produced a 2% absolute improvement on the training questions.

2.2 Confidence Estimation

We used three features to estimate $P(correct|Q,A)$. One feature is whether the answer satisfies the verb arguments of the question. This is a Boolean feature and we denote it VS. Using the training questions, we obtained $P(correct|VS$ is $true)$=0.49 and $P(correct|VS$ is $false)$=0.23, which clearly indicate VS is predictive of answer correctness.

The second feature is a pair of integers (m, n), where m is the number of content words in common between the question and the context of the answer, and n is the total number of content words in the question. The answer context is a text window that is 30 word wide and has the answer at the center. Table 1 shows $P(correct|m, n=4)$ computed from training questions. As expected, $P(correct|m, n)$ monotonically increases with m when n is fixed.

	m=0	m=1	m=2	M=3	m=4
$P(correct\|m,n=4)$	0.10	0.10	0.12	0.19	0.34

Table 1: $P(correc|m,n)$ when n=4, computed from training questions.

The third feature is T, the answer type of the question. Table 2 shows $P(correct|T)$ as computed from the training questions. As expected, some types of questions (e.g. Person) are more likely to result in a correct answer than other types of questions (e.g. Animal).

T	$P(correct\|T)$
Location	0.24
Person	0.39
Date	0.26
Quantity	0.21
Cardinal Number	0.35
Organization	0.36
Animal	0.0
Misc	0.14

Table 2: $P(correct|T)$, computed from training questions. Types with too few training questions were put into the Misc category.

Since we do not have enough training data to directly estimate $P(correct|VS, m, n, T)$, we computed $P(correct|Q, A)$ by fitting $P(correct|VS)$, $P(correct|m,n)$ and $P(correct|T)$ in a mixture model:

$$\begin{aligned} P(correct|Q, A) &\approx P(correct|VS, m, n, T) \\ &\approx P(correct|VS) \times 1/3 + P(correct|m,n) \times 1/3 + P(correct|T) \times 1/3 \end{aligned}$$

Since the parameters were pre-computed from the training questions, the computing of $P(correct|Q,A)$ for new Q-A pairs requires only a few table lookups.

3. Using the Web for QA: BBN2002B and BBN2002C

Some studies reported positive results using the World Wide Web to supplement the TREC corpora for question answering (Dumais et al, 2002). The idea is simple: the enormous amount of data on the Web makes it possible to use very strict, precision oriented search criteria that would be impractical to apply on the much smaller TREC corpora.

Our technique to exploit the Web for QA is similar to Dumais et al's. We used the Web search engine Google because of its efficiency and coverage. Similar to (Dumais et al, 2002), we used two forms of queries, exact and non-exact. The former rewrites a question into a declarative sentence while the latter is a conjunction of all content words in the question. For efficiency considerations, we only looked for answers within the top 100 hits for each Web search. Furthermore, we confined to the short summaries returned by Google rather than using the whole Web pages in order to further cut the processing cost. The summaries were processed using BBN's IdentiFinder. The most frequent named entity that matches the answer type of the question was extracted as the answer. The QA guideline requires the ID of a document in the TREC corpus that supports the answer. The highest ranked document that contains the answer string was used for that purpose.

Both BBN2002B and BBN2002C used the Web for QA, but they used different methods for confidence estimation. For BBN2002B, the confidence of an answer found from the Web is a function of the type of the question and the frequency of the answer in the Google summaries. Specifically,

$$P(correct|Q, A) \approx P(correct|T, F) \approx P(correct|T) \times 0.5 + P(correct|F) \times 0.5$$

where
T = question type
F = frequency of A in the Google summaries

For BBN2002C, the confidence of an answer A is a function of its frequency F in the Google summaries and a Boolean variable $INTREC$, which is true if and only if A was also returned by the base run (BBN2002A) from the TREC corpus. Specifically,

$$P(correct|Q, A) \approx P(correct|F, INTREC)$$

The Boolean variable *INTREC* is a useful feature because a lot of data on the Web is of dubious quality and as such some kind of validation of the Web answers is necessary. Figure 1 plots the probability of answer correctness as a function of F and *INTREC*. The figure shows that there is a strong correlation between F and answer correctness. The probability of answer correctness also strongly depends on the Boolean feature *INTREC*.

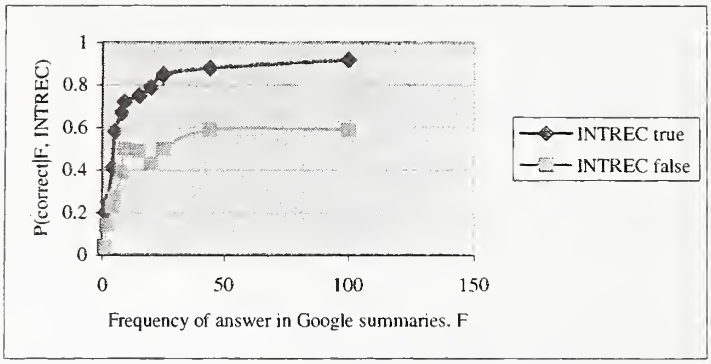


Figure 1: *P(correctness | F, INTREC)*, **computed from training questions**

The question-answer pairs from the Web were merged with the ones produced by the base run (i.e. BBN2002A). Since for the TREC2002 QA track each question can only have one answer, we chose the one with the higher confidence score if the Web answer and the answer from base run are different for a question. If the Web answer and the answer from the base run agree, the confidence score took the maximum of the two.

4. TREC2002 Results

We measured our TREC 2002 QA results using two scores. The first is the un-weighted score, which is the percentage of questions for which the answer is correct. The second is the confidence-weighted score, as described by Voorhees, 2003. Although the confidence-weighted score does not directly reflect the goodness of the confidence estimation, they correlate strongly because the score rewards systems that place correct question-answer pairs ahead of incorrect ones. It is easy to verify that the un-weighted score is a baseline for the confidence-weighted score where the confidence estimation (and as a result the order of the question-answer pairs) is completely random. Therefore, one way to determine how well a confidence estimation method works is to compare the two scores.

Table 3 shows the results of our three runs. Two observations can be made. First, the Web-supplemented runs (BBN2002B and BBN2002C) are significantly better than the base run (BBN2002A), confirming findings published in earlier studies (Dumais et al, 2002). Second, our confidence estimation techniques work reasonably well: The confidence-weighted score is significantly better than the un-weighted score for all three runs. This is especially true for BBN2002C, where the difference between the weighted and the un-weighted scores is rather small.

	Un-weighted score	Confidence-weighted score	Upper-bound of confidence-weighted score
BBN2002A	0.186	0.257	0.498
BBN2002B	0.288	0.468	0.646
BBN2002C	0.284	0.499	0.641

Table 3: Un-weighted, confidence-weighted and upper-bound scores for BBN2002A, BBN2002B and BBN2002C.

5. Conclusions

We described our question answering work for the TREC2002 QA track. In particular, we have explored two problems: answer selection and confidence estimation. We found that some simple constraints can improve a pure IR-based technique for answer selection. Our confidence estimation techniques used a few simple features such as question type, verb argument satisfaction, the number of question words matched by the answer context and the answer frequency in the retrieved Web pages. Performance scores show that our confidence estimation techniques work reasonably well. Our results also confirmed findings by other researchers that the Web is a useful resource for answering TREC-style factoid questions.

References:

M. Kaszkiel and J. Zobel, "Effective Ranking with Arbitrary Passages", Journal of the American Society for Information Science (JASIS), Vol 52, No. 4, February 2001, pp 344-364.

S. Miller, H. Fox, L. Ramshaw, and R. Weischedel. 2000. A Novel Use of Statistical Parsing to Extract Information from Text. In *Proceedings of the North American Association for Computational Linguistics.*

D. Miller, T. Leek, and R. Schwartz, 1999. “A hidden markov model information retrieval system.” In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1999.

S. Dumais, M. Banko, E. Brill, J. Lin and A. Ng. “Web Question Answering: Is More Always Better?”. In Proceedings of ACM SIGIR 2002.

D. Bikel, R. Schwartz, R. Weischedel, "An Algorithm that Learns What's in a Name," Machine Learning, 1999.

E. Voorhees, 2003. “Overview of the TREC 2002 Question Answering Track.” In TREC 2002 Proceedings.

TREC 2002 Cross-lingual Retrieval at BBN

Alexander Fraser[1], Jinxi Xu and Ralph Weischedel
BBN Technologies
50 Moulton Street
Cambridge, MA 02138

1. Introduction

We used basically the same retrieval system we used in TREC 2001. Our experiments featured a different method for estimating general English probabilities, two additional Arabic stemmers, a more complex model for lexicon extraction from parallel texts and a slightly different method for query expansion. To our disappointment, these changes did not improve retrieval performance.

1.1 Retrieval System

Our retrieval system was documented in (Xu et al, 2001). It ranks documents based on the probability that a query is generated from a document:

$$p(Q \mid D) = \prod_{t_q \text{ in } Q} (\alpha P(t_q \mid GE) + (1-\alpha) \sum_{t_d \text{ in } D} p(t_d \mid D) p(t_q \mid t_d))$$

$$p(t_d \mid D) = \frac{\textit{frequency of } t_d \textit{ in } D}{\textit{size of } D}$$

Where Q is a query, D is a document, t_q's are query terms, t_d's terms in the document. The mixture weight α is fixed to 0.3.

Two sets of parameters are important in the retrieval model. One is the translation probabilities $P(t_q|t_d)$. In TREC 2001, we used model 1 of Brown's statistical MT work (Brown et al, 1993) for estimating term translation probabilities from a parallel corpus due to efficiency considerations. With more computer power at disposal, for TREC 2002 we used the more complex but potentially more accurate model 4 for the same purpose. Differences between the two models were discussed by Brown et al, 1993.

The other is the general English probabilities $P(t_q|GE)$, which model the importance of the query terms for retrieval. In TREC 2001, we used a large English corpus (news stories in TREC English vols 1-5) for estimating $P(t_q|GE)$, by dividing the frequency of the term by the size of the English corpus, based on the assumption that the English and the Arabic corpora are sufficiently close in content and genre. This assumption is clearly not true. The English corpus consists of stories published in the late 80's and early 90's while the Arabic AFP corpus consists of articles published in the late 90's and 2000. Second, the two corpora focus on different geographic regions (AFP on Middle East

[1] Alexander Fraser is currently with Information Sciences Institute, University of South California

while TREC English on U.S.). Since finding a closely matched English corpus for AFP is hard, for TREC 2002 we computed GE probabilities based on the statistics of the Arabic translations of the English terms, using a technique proposed by Hiemstra et al, 1999:

$$p(e \mid GE) = \sum_{\text{Arabic words } a} p(e \mid a)p(a \mid GA)$$

where $p(a|GA)$ was computed based on the frequency counts of the Arabic terms in the AFP corpus.

1.2 Lexical Resources

We used two lexical resources for term translation, a parallel corpus and a manual lexicon. The parallel corpus was obtained from the United Nations (UN). The United Nations web site (http://www.un.org) publishes all UN official documents under a document repository, which is accessible by paying a monthly fee. A special purpose crawler was used to extract documents that have versions in English and Arabic. After a series of clean-ups, we obtained 38,000 document pairs with over 50 million English words. For sentence alignment, a simple BBN alignment algorithm was used. Translation probabilities were obtained by applying a statistical machine translation toolkit, GIZA++ (Och and Ney, 2000) on the UN corpus. GIZA++ is based on the statistical translation work pioneered by (Brown et al, 1993). We experimented with both model 1 and model 4 for lexicon extraction. The manual lexicon was obtained from Tim Buckwater (Buckwalter, 2001). It contains about 86,000 word pairs.

1.3 Arabic Stemmming

In TREC 2001 CLIR, we used the Buckwalter stemmer (Buckwalter 2001) for stemming Arabic words. It is table-driven, employing a number of tables that define all valid prefixes, stems, suffixes, and their valid combinations. Given an Arabic word *w*, the stemmer tries every segmentation of *w* into three sub-strings, $w=x+y+z$. If x is a valid prefix, y a valid stem and z a valid suffix, and if the combination is valid, then y is considered a stem. We modified the stemmer so that it only stems a word if the word has exactly one possible stem. Otherwise, the original word is returned. The performance of the Buckwalter stemmer depends on the coverage of the stem table: Words whose stems are not in the stem table cannot be stemmed by the stemmer.

In TREC2002, we experimented with two new Arabic stemmers as well as Buckwalter. One is UMass Light 8 (Larkey et al, 2002). The other is Al-Stem (Darwish, 2002), the standard stemmer for TREC 2002 CLIR. Both are rule-based and as such are not affected by lexicon coverage. Recent studies (Larkey et al, 2002; Darwish and Oard, 2002) demonstrated that rule-based stemmers are suitable for Arabic retrieval.

1.4 Query Expansion

In our TREC 2001 experiments, English and Arabic query expansions were performed sequentially: We performed English expansion first and then used the expanded English queries to retrieve the top documents for Arabic expansion. A potential problem with sequential expansion is that it can propagate errors made in the English expansion to the

Arabic expansion. In TREC 2002, we experimented with parallel expansion: We performed English and Arabic expansions independently, using the original unexpanded queries in the initial retrieval for expansion of both languages.

For English query expansion, we used a corpus of 1.2 million articles from sources AP, Reuters and FBIS. For Arabic query expansion, we used the AFP corpus and optionally additional articles from two newspaper sources Al-Hayat and An-Nahar. The expansion parameters are identical for both languages (English and Arabic): 50 terms were selected from 10 top retrieved documents based on their total TF.IDF in the top documents. The expansion terms and the original query terms were weighted as follows:

$$weight(t) = old\_weight(t)+0.4*\sum TFIDF(t, D_i)$$

where D_i's are the top retrieved documents.

1.5 Spelling Normalization

We used the same procedure we used last year to normalize spelling variations in Arabic words. Two kinds of spelling variations were considered. The first is the confusing of the letter YEH (ي) and the letter ALEF MAKSURA (ى) at the end of a word. Since variations of this kind usually result in an "invalid" word that is un-stemmable by the Buckwalter stemmer, our solution is to detect such "errors" using the stemmer and restore the correct word ending. The second is to write diacritical ALEFs (e.g. أ , إ and آ) as the plain ALEF (ا). In our experiments, we replaced all occurrences of the diacritical ALEFs by the plain ALEF.

2. Results of Submitted Runs

We submitted four runs—all are cross-lingual runs. The runs differ in the following aspects:

- The model used for lexicon extraction from the parallel corpus, *model 1 vs model 4*
- The lexical resources used for term translation
- The Arabic stemmer(s) used
- The Arabic corpus used for query expansion
- The query expansion method, *sequential vs parallel*
- The method the GE probabilities was calculated, *old vs new*. The old method computed the GE probabilities from the TREC English corpus while the new method computed them from the Arabic AFP corpus.

Table 1 shows the features of our submitted runs. BBN11XLS is our standard resource run. BBN11XLC essentially repeated our TREC 2001 work on the TREC 2002 query set. To our disappointments, the changes we made to last year's work did not produce better retrieval results, as shown by Table 2. In fact, the collective effect of the changes is a noticeable degradation in the retrieval performance (BBN11XLA and BBN11XLB vs

BBN11XLC). We are currently analyzing the impacts of the individual changes on retrieval.

	Model for lexicon extraction	Lexical resources	Arabic stemmer	Arabic Expansion Corpus	Query expansion	GE probabilities
BBN11XLA	Model 4	Parallel corpus and manual lexicon	Buckwalter and UMass Light 8	AFP. Al-Hayat. An-Nahar	Parallel	New
BBN11XLB	Model 4	Parallel corpus and manual lexicon	UMass Light 8	AFP. Al-Hayat. An-Nahar	Parallel	New
BBN11XLC	Model 1	Parallel corpus and manual lexicon	Buckwalter	AFP. Al-Hayat. An-Nahar	Sequential	Old
BBN11XLS	Model 1	Parallel corpus	Al-Stem	AFP	Parallel	New

Table 1: Description of sumbitted runs for TREC 2002 CLIR. BBN11XLA used two stemmers: Buckwalter for term translation and UMass Light 8 for stemming the Arabic expansion terms.

	BBN11XLA	BBN11XLB	BBN11XLC	BBN11XLS
Average Precision	0.3444	0.3514	0.3756	0.3473

Table 2: Retrieval results of submitted runs

Acknowledgements

This research is sponsored by the Defense Advanced Research Projects Agency and managed by SPAWAR Systems Center under contract N66001-00-C-8008. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied of the Defense Advanced Research Projects Agency, the SPAWAR Systems Center, or the United States Government.

References

P. Brown, S. Della Pietra, V. Della Pietra, J. Lafferty and R. Mercer, 1993. "The Mathematics of Statistical Machine Translation: Parameter Estimation". In *Computation Linguistics,* 19(2), 1993.

T. Buckwalter, 2001. Personal Communications.

K. Darwish. http://www.glue.umd.edu/~kareem/research/.

K. Darwish and D. Oard, 2002. "Term Selection for Searching Printed Arabic." In ACM SIGIR 2002.

Hiemstra, D. and de Jong, F. 1999. "Disambiguation strategies for cross-language information retrieval." In Proceedings of the third European Conference on Research and Advanced Technology for Digital Libraries, pages 274-293, 1999.

L. Larkey, L. Ballesteros and M. Connell, 2002. "Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis." In ACM SIGIR 2002.

F. Och and H. Ney, 2000. "Improved Statistical Alignment Models." In proceedings of *ACL 2000*.

J. Xu, R. Weischedel and C. Nguyen, 2001. "Evaluating a Probabilistic Model for Crosslingual Retrieval." In proceedings of ACM SIGIR 2001.

Information Filtering, Novelty Detection, and Named-Page Finding

Kevyn Collins-Thompson, Paul Ogilvie, Yi Zhang, and Jamie Callan
Language Technologies Institute
Carnegie Mellon University
{kct,pto,yiz,callan}@cs.cmu.edu

1. Introduction

In TREC 11, our group participated in the Novelty track, Filtering track, and the Named-Page Finding task of the Web track. This paper describes our approaches, experiments, and results. As the approach for each task is quite different, the paper contains a section for each of the tasks. The following section describes our experiments in adaptive filtering, Section 3 describes named-page finding, and section 4 discusses the Novelty track.

2. Adaptive Filtering

In the adaptive filtering track, we used the same system as in TREC9 and TREC10. The Rocchio algorithm is used for anytime profile updating. More detailed information about profile updating and the system structure is available in [17]. This year, we focused on comparing different thresholding methods, and also did some experiments on using language model to improve initial query profiles.

2.1 Thresholding

Two evaluation measures were used in this year's adaptive filtering track: T11U=2*R_+-N_+[1] and T11F=1/(1/*recall*+4/*precision*)), where R_+ is the number of relevant documents delivered, and N_+ is the number of non-relevant documents delivered. T11U can be optimized if we can estimate precision, and the corresponding optimal strategy is: deliver if P(*relevant*) > 0.33. T11F can be optimized only if we can estimate both precision and recall.

When filtering, we have several training documents represented as $((x_1,y_1)(x_2,y_2),...(x_t,y_t))$, where x_i is the score of a delivered document with user feedback, and $y_i=0$ if document i is not relevant, otherwise $y_i=1$. We also order the tuples according to the constraint $x_1<x_2<...x_i<x_{i+1}<...<x_t$.

	Optimized for T11U	Optimized for T11F
NE	Yes	Yes
ML	Yes	Yes
Empirical Optimal	Yes	No
Logistic Regression	Yes	No
Bayesian Error Model	Yes	No
Greedy Search	Yes	No

Table 1: Candidate Thresholding Algorithms

We tried six threshold-setting algorithms (Table 1):

- NE (Normal-Exponential): Model the scores of the relevant documents with a Normal distribution and model the top ranking non-relevant documents with an exponential distribution as described in [1].
- ML: Maximum Likelihood Estimation as described in [19]. Use the same model as NE, but explicitly model the sample bias while estimating model parameters. This is a modified version of the Maximum Likelihood Estimation thresholding algorithm, because the early stage (when the number of relevant documents or non-relevant documents in the training set is smaller than 4, the threshold is the optimal one calculated by the model, smoothed with the old threshold and the average relevant document score using linear interpolation.
- EO (Empirical Optimal). Let all candidate threshold be $\theta_i=x_i+x_{i-1}/2$ $(i=1\ldots t)$. Set the real threshold at θ_j, where the evaluation measure we want to optimize achieved the empirical optimal value on training data among all of the candidate points.
- Logistic Regression: This is a strict implementation of widely known logistic regression algorithm. Although the thresholding algorithm described in [12] is also based on logistic regression, it is a modified version using calibration to fit the filtering task and data, and would probably get a better result.

[1] The exact official evaluation for utility is T11SU=(max(T11U/MaxU, MinNU)-MinNU)/(1-MinNU), where MaxU=2*(Total Number of relevant documents). MinNU=-0.5. T11SU is normalized T11U.

- Bayesian Error Model: Use uniform error model $P(y=1|x,t,e)=e+(1-2e)\Phi(x-t)$, where $\Phi(x-t)=0$ if $x<t$, and $\Phi(x-t)=1$ if $x>=t$. The system use a Beta distribution $p(e) \sim \beta(\alpha_1,\alpha_2)$ to model the prior of error e. Bayesian estimation of $P(y=1|x, \textit{training data})$ is used to estimate the precision. More information about this is available in [1].
- Greedy Search: This is a greedy algorithm that increases the threshold if a relevant document is delivered, while decrease the threshold if a non-relevant document is delivered. While the step size depends on 1) the difference between the score of the document and the current threshold and 2) how many changes the system has made before. So the strategy will modify the threshold if given a feedback for document d_i using: $t_{new}=t_{old}-2\delta_1*\max(\delta_2,score_{di}-t_{old})$ if d_i is relevant, otherwise $t_{new}=t_{old}+\delta_1*\max(\delta_2,score_{di}-t_{old})$. Where δ_1 decreases as number of feedback increases, δ_2 is set to 0.005 arbitrarily.

Notice that Logistic Regression and the Bayesian Error Model are focused on model $P(y|x)$ and do not model the marginal distribution $P(x)$. These two models can help estimate precision, but are not capable of estimating the recall. Theoretical optimization for T11F is not possible for them without getting an estimate of $P(x)$. This is a general problem for using discriminative models for thresholding, and one solution is to use the empirical distribution of x as described in [12] to help estimate recall, but it requires a lot of computation.

	TREC8	TREC9	TREC11
NE	0.324	0.360	0.365
ML	0.340	0.363	0.373
Empirical Optimal	0.213	0.344	0.315
Logistic Regression	0.212	0.300	0.289
Bayesian Error Model	0.268	0.303	0.311
Greedy Search	0.051	0.155	0.090

Table 2: T11U performance with different thresholding algorithms.

We compared these different thresholding algorithms on TREC8 and TREC9 filtering track data, using T11U as the evaluation measure. Maximum Likelihood Estimation works consistently the best on both data sets. The system was tuned using the TREC8 and TREC9 filtering track data. After submitting our results, we did more experiments on TREC11 data. The final results on the different datasets are shown in Table 2. Using the Normal and Exponential model to model the score distribution worked well on all three dataset, although their performance was not good on TREC10 data(not reported here. [18]).

2.2 Modifying an initial profile without training documents

NIST provides topics that contain title, description and narrative fields to describe the user's information interests, but what should be the initial profile description is unknown. If we look at the following sample topic from TREC8:

Query: q353

Title: Antarctica exploration

Description: Identify systematic explorations and scientific investigations of Antarctica, current or planned.

Narrative: Documents discussing the following issues are relevant: - systematic explorations and scientific investigations of Antarctica (e.g., seismology, ionospheric physics, possible economic development) - other research currently conducted or planned for the future - banning of mineral mining Documents discussing tourism are non-relevant. Documents discussing "disrupting scientific experiments" are non-relevant unless a specific experiment is identified.

We can see that the title is a better description but contains only 2-4 words. The Description and the Narrative are too long and noisy. We tried a mixture model to model how a user generates a description/narrative. Assuming a query is generated by a mixture of profile independent *general query model* M_g and profile dependant *core query model* M_q. Words such as "Identify" or "Documents" are more likely to be generated by M_g, while the words "Antarctica" is more likely to be generated from M_q. Using algorithms described in [21], we can find the *core query model* M_q and use it to do feature selection or reweighing for the initial query. Thus for each profile, we have 7 options for setting initial queries based on how we use title, description and narrative fields provided by NIST. The results on the TREC8 and the TREC11 dataset are shown in Table 3. Using the mixture model helped for the TREC8 dataset, while we didn't see significant improvement on TREC11 dataset.

Initial query	TREC8	TREC11
Title	0.3185	0.3621
Title + Description	0.3297	0.3732
Title + Description + Mixture model for reweighing words	0.3405	0.3736
Title + Description + Mixture model for feature selection	0.3524	0.3668
Title + Description + Narrative	0.3559	0.3669
Title + Description + Narrative + Mixture model for reweighing words	0.3424	0.3741
Title + Description + Narrative + Mixture model for feature selection	0.3402	0.3673

Table 3: T11U performance with different initial profile settings.

		UDESC	FDESC	Uml	Fml
= Max		0	1	0	0
> Med		97	90	92	88
< Med		3	10	8	12
Topic 1-50	T11U	0.445	0.431	0.447	0.433
	T11F	0.422	0.401	0.410	0.396
Topic 51-100	T11U	0.291	0.293	0.290	0.292
	T11F	0.041	0.038	0.034	0.035

Table 4: Performance of our official runs.

2.3 Filtering Track Results

This year, we submitted four very similar runs. UDESC and FDESC used description and title fields, while Uml and Fml used the mixture model for term reweighting. Table 4 shows our results compared with other systems. For most topics, our system performance is above the median performance.

Figure 1 shows the results for one of the 4 runs on each topic, compared with Max, Median and Baseline (the performance of a system that delivers nothing). The system performance on the first 50 topics is much better than the performance on the last 50 topics. The first 50 are created and annotated by NIST annotators, while the last 50 topics were intersections of Reuters categories. Figure 2 compares the first 50 topics with the last 50 topics by sorting the topics according to the number of relevant documents in that topic. Although the number of relevant documents is similarly distributed for the first 50 and the last 50 topics, the best performance is quite different. When we look at some relevant documents for NIST annotated topics and "intersection" topics, we feel it is hard to learn "intersection" profiles according to training documents with a bag of word model (Rocchio). More detailed research and analysis is needed to fully understand the difference between them.

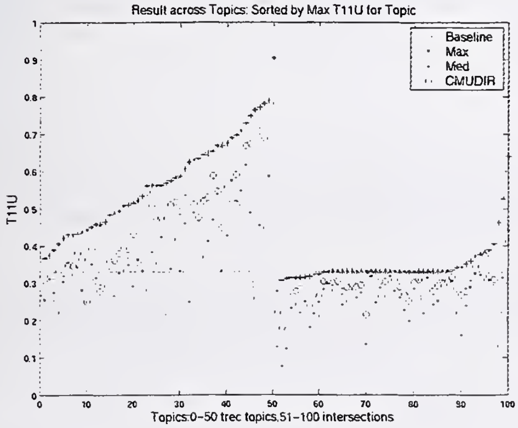


Figure 1: Compare our result (CMUDIR) with the best performance (Max) and the Median Performance (Med) for each Topic.

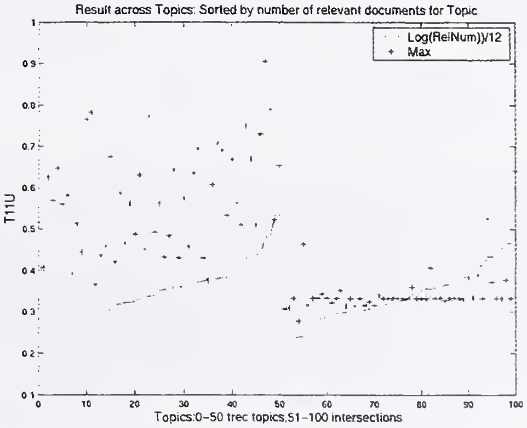


Figure 2: TREC annotators create the first 50 topics, and the last 50 topics are created by intersection of Reuter's categories.

3. Named-Page Finding

The language modeling approach to information retrieval typically makes the assumption that the query is representative of the relevant documents. In most previous research, language modeling places equal weight on all parts of the document. This may work well for ad-hoc document retrieval on newspaper corpora, but we do not feel that this necessarily makes the best use of information present in the document. For example, it does not leverage document structure, such as markup present in HTML documents. For named-page finding we hypothesize that the user's query is what the user believes to be a reasonable estimate of the "name" of the page she is seeking. Therefore, when estimating a language model, we do not want to estimate the language model of the entire document, but instead we want to estimate a model for the page's "name". Given some document structure, we can form a variety of document representations. We can weight these representations according to how characteristic they are of the page's "name". For example, we may want to weight the title of a document more than the rest of the text. Or in a hypertext environment, such as the Internet, we may want to incorporate the text of the links pointing to the page for which we are estimating a language model.

Language modeling suggests that we try to estimate a new language model from language models created from the document representations. This new language model for a document should be designed so that it closely models what we would expect a user to write as a query when requesting the document. For named-page finding, we would like to estimate a language model for the "name" of the page from language models produced by various document representations. We explore creating this new language model by taking a linear interpolation of the other models. Note that this is different from doing a linear combination of scores from different systems: we directly estimate the probability of a word given the differing language models. This is also different from directly weighting the term frequencies. Isolating the fields into different language models allows for smoothing each representation with a collection language model based on that representation only. This explicitly models the fact that the language usage in different document representations or fields is different, and adjusts the probabilities accordingly.

As mentioned above, we form language models from different document representations of the document. For example, one representation of a document may be its title. Another document representation may consist of the text contained in larger fonts. These representations do not need to be partitions, or even non-overlapping. We can still use the entire content of a document as a representation, while including other representations such as the document's title. From these different representations, we form a language model, using like representations from other documents for the backoff language model. We combine the language models from different representations using linear interpolation to form a new language model. The representations used and the linear interpolation weights chosen can be fine-tuned to a specific task. For example, we may expect that the document's title is more important for named-page finding than it is for a general ad-hoc relevance task. If so, then for named-page finding we would use a higher weight for the language model formed from the title than when performing an ad-hoc relevance search.

In this report, we investigate combining document representations to improve retrieval performance for named-page finding. We also explore how much information is needed to achieve performance that is similar to using the full document and in-link text available.

3.1 Language Modeling for Named-Page Finding

Two primary models for ranking documents are used in Information Retrieval: Kullback-Leibler divergence [15] and the generative language model [10]. Under common assumptions we make for this task, these approaches are equivalent [9]. So we arbitrarily choose to use a generative language model.

$$\mathrm{P}(Q|D) = \mathrm{P}(Q|\theta_D) = \prod_{w \in Q} P(w|\theta_D)$$

In the above equation, Q is a query (a sequence of words), D is a document, and θ is a language model. For the generative model, the query is treated as a sample from the document's language model, and we must then compute the probability that the document's language model produced the query sequence. Note that the probability of the term is in the product as many times as the term occurs. That is, we represent the query as a sequence of words.

To complete the specification of the retrieval method described above. we need to estimate the language models. The methods we use here are discussed and compared in [16]. The simplest way to estimate a unigram language model given a chunk of text is to use a maximum likelihood estimate:

$$P_{MLE}(w|\theta_T) = \frac{count(w;T)}{|T|}$$

Here, T denotes the text we are using to estimate the language model. The probability of the word given the text's model is simply the number of times the word occurs in the text divided by the length of the text. This has the advantage of being easy to compute, but it has the problem that many words have zero probability or are poorly estimated if the length of the text is small. This technique is often used to estimate a background language model, such as the probability of a word given the entire collection. The collection is typically large enough to give estimates. To address the problem zero counts, linear interpolation is often used.

$$P_{LIN}(w|\theta)=\sum_{i=1}^{k}\lambda_i P(w|\theta_i)$$

Where k is the number of language models we are combining, and λ_i is the weight on the model θ_i. To ensure that this is a valid probability distribution, we must place these constraints on the lambdas:

$$\sum_{i=1}^{k}\lambda_i = 1 \quad \text{and for } 1 \le i \le k,\ \lambda_i \ge 0$$

We often use a linear interpolation of a text model and a collection model. One specific form of this is to use Dirichlet prior smoothing. This technique has worked quite well in ad-hoc retrieval experiments [16][9][14]. Dirichlet prior smoothing has one parameter, μ, which is typically chosen close to the average length of the text chunks being estimated. For this smoothing method, we have θ_1 as the text model, θ_2 as the collection language model, $\lambda_1 = |T|/(|T|+\mu)$, and $\lambda_2 = \mu/(|T|+\mu)$.

A final technique we will use to estimate a language model is a character based n-gram model. An n-gram model considers the context of the token. Specifically, it estimates the probability of the token given the previous n - 1 tokens.

$$P(c_i|c_{i-n+1}c_{i-n+2}\ldots c_{i-1},\theta_T)=\frac{P(c_{i-n+1}c_{i-n+2}\ldots c_{i-1}c_i|\theta_T)}{P(c_{i-n+1}c_{i-n+2}\ldots c_{i-1}|\theta_T)}$$

The n-gram model does not specify how the probabilities on the right hand side of the above equation should be estimated, but they can be estimated using any of the previously described techniques. Using backoff where the background models are based on shorter sequences of tokens is a common method for estimating these probabilities.

3.2 System Specifics

We use the Lemur toolkit [7] for document indexing and retrieval. For document tokenization we used Inquery's stopword list and the Porter stemmer. The URLs were tokenized on punctuation (., /) and were not stemmed. A shorter stopword list was used for URLs ("http", "www", "com", "gov", "html", etc.). Each document had as many as seven document representations, outlined in Table 2. For every representation except the URL, we formed language models using a backoff model with Dirichlet prior smoothing. The Dirichlet prior parameter was chosen to be close to twice the average length of the representation. The probability of a word given the document's URL was computed treating the URL and word as a character sequence, then computing a character-based trigram generative probability. The numerator and denominator probabilities in the trigram expansion were estimated using a linear interpolation with the collection model (all URLs in the corpus).

The linear interpolation parameters for the .GOV corpus were trained using 80 queries we created locally for the named-page finding task. We trained the lambda parameters by performing retrieval separately on each of the representations. The weights were then taken as the scaled mean reciprocal rank of the system. The normalization was performed to ensure that the weights summed to one. This training procedure did not yield better results than assigning equal weight to each representation.

3.3 Named-Page Finding Results

We briefly describe experiments on the WT10G testbed (Table 5). Without the use of document priors. our system has respectable performance. This technique is as strong as any reported in TREC10 that did not use a form of document prior [3]. The best performing single document representation. document in-link text, had a MRR of 0.515, so combining the document representations significantly improves performance. Additionally, we tried using the document URL priors described in [4] as a re-ranking strategy for the top 1000 documents. The use of document priors did improve performance for all evaluation measures used for the task.

Configuration	MRR	% TOP 10	% FAIL
Equal lambdas	.676	83.4	5.5
Equal lambdas + URL length prior	.799	91.7	3.4

Table 5: Results of the homepage finding task on the WT10G testbed

Representation	Description	MRR	% TOP 10	% FAIL
Alt	Image alternate text	.194	28.0	66.7
Font	Changed font sizes and headings	.191	25.3	68.0
Full	Full document text	.469	66.7	16.7
Link	In-link text	.455	58.0	32.0
Meta	Meta tags (keyword, description)	.144	21.3	75.3
Title	Document title	.407	56.0	35.3
URL	Character trigram on URL	.131	19.3	68.7

Table 6: Performance of individual document representations on the named-page finding task

Run	Configuration	MRR	% TOP 10	% FAIL
LmrAllEq	Alt+font+full+link+meta+title+url (equal parameters)	.676	88.0	3.3
LmrAllEst	Alt+font+full+link+meta+title+url (trained parameters)	.667	86.7	3.3
LmrSmall	Link+font+title	.589	73.3	16.7
LmrDocStruct	Alt+font+full+meta+title	.567	72.7	15.3
LmrNoStruct	Full+link	.611	84.0	8.7

Table 7: Official results of the named-page finding task on the .GOV testbed

For this year's TREC, we wished to investigate two main questions: whether we get good performance gains by combining document representations, and what performance we can get when operating under a variety of system constraints. First we look at the performance of the individual document representations. This is in Table 6. The full text, in-link text, and title text were the best document representations for the named-page finding task; full text yielded the greatest performance with a mean-reciprocal rank of .469.

Our official results are summarized in Table 7. Two of our official submissions combined all of the representations: LmrAllEq and LmrAllEst, which respectively had mean-reciprocal ranks of .676 and .667. LmrAllEq used equal weighting parameters and LmrAllEst used parameters from our naïve training procedure evaluated on our 80 query training set. Both runs had much better performance on all measures than any of the individual methods. LmrAllEst performed slightly worse than LmrAllEq, but we are not sure whether this difference is significant.

The other three runs investigated combining document representations under different scenarios. LmrNoStruct used only the full text and the link text. The name is a little misleading, as it did keep the full text and link text separate as different language models. Combining these two best representations yielded a MRR of .611, which is not quite as good as combining all of the document representations. LmrSmall was an attempt to estimate what level of performance can be maintained while indexing only small amounts of text. Only the font, title, and link representations were used for this run. This resulted in indexing only 43 million terms, where the full text index contains around 945 million terms. The MRR of .589 for this run was not bad, but the failure rate at 50 documents was quite high, and the number of topics with the right answer in the top 10 was also lower than for other runs. The other run LmrDocStruct, looked at document representations from the document only, ignoring the URL and the in-link text. Incremental indexing of in-link text can be a complicated operation, and it may be too expensive to scan the entire corpus on a regular basis to compute each document's in-link text. This run was an attempt to measure how well a system could perform without the use of this information. The MRR for this run was .567, which suggests that in-link text is an important document representation for named-page finding.

3.5 Conclusions

We explored the use of document structure in the named-page finding task and the homepage finding task. We found that combining different document representations worked very well within the language modeling framework. We feel that the use of language modeling provides an effective mechanism for combining information from different document representations. We found that document in-link text is important for named-page finding, confirming previous results in homepage finding. We also demonstrated that good MRR performance can be achieved with a very small index, but that in order to get a high percentage of documents found in the top ten answers and to preserve a low failure rate, the full text of the document is needed. We also showed that by adding more representations, we can improve performance, even though the content of the new representations may overlap with other existing representations.

The largest issue that we failed to adequately address was the training of the linear interpolation weights for the combination of document representation language models. The training method we used did not seem to provide any gain. We would like to explore more sophisticated techniques for training the parameters in the future.

4. Novelty Track: Finding Relevant and Redundant Sentences

The problems of finding relevant and redundant sentences are related to several other well-studied IR problems, with important differences. Finding specific relevant sentences is similar to some aspects of open-domain question-answering, in that we are looking for specific statements, and not just entire documents, that satisfy the query. We therefore include more surface features of sentences, such as punctuation and named entity types, in our analysis. However, the nature of the answers is much less specific than that typically seen in open-domain QA applications. Another similar problem is that of topic-level novelty and redundancy detection, as described by Zhang et al. [20], in which the authors use statistical models to perform adaptive filtering to find documents that are not only relevant, but also novel (or equivalently, not redundant). The novelty problem is also related to multi-document, query-specific summarization, in that we seek to find a set of representative, maximally informative sentences. However, the criteria for "maximally informative" are different for each problem: summarization seeks to obtain good coverage of the various aspects of the relevant information while keeping within a size/length constraint. For our problem, we have no length constraint and the definition of "novel" is extended to allow more subtle differences between sentences.

Our general approach is to view both relevant and redundant sentences as simple statistical translations of the query. For performance reasons, we currently only apply this type of model to the redundant sentence computation, and use a tf.idf-based approach to obtain relevant sentences.

4.1 Finding Relevant Sentences

We examined the performance of tf.idf-based retrieval using sentences as "documents" and found that, with pseudo-relevance feedback, using all sentences with a non-zero score gave high recall of relevant sentences (typically 70-80%). However, not surprisingly, the precision was extremely low (usually 10% or less), so this led to the following method.

1. Retrieve a set of candidate sentences using straightforward tf.idf-based retrieval with query expansion, based on a query constructed from the TREC description
2. Extract a set of features from the resulting candidate sentences
3. Use these features to classify each candidate sentence and remove those that are more likely to be non-relevant.

The classification in step 3 can be based on one of several different methods. In this study, we examined these three:

Simple_Threshold: use the tf.idf score as the only feature, and apply a threshold;
Decision_Tree (DT): extract a much wider set of features and build a decision tree; and
Proximity: simple model using proximity to highly relevant sentences as the main criterion.

We treated each sentence as a separate document, and indexed all the sentences from all relevant documents using Lemur [7]. We created a query from the title, description, and narrative fields in the topic and use this to score each sentence using tf.idf weighting. We performed query expansion using pseudo-relevance feedback, using the top 10 terms from the top 20 sentences. This produced an initial "candidate list" of sentences.

Simple_Threshold Rule

The tf.idf scores were normalized so that the highest scoring sentence received a score of 1.0. This rule was mostly useful as a baseline, and as a way for finding highly relevant sentences for the Proximity rule.

Decision Tree Rule

In this approach, we extracted a "base set" of about 15 to 20 surface and semantic features from a document sentence. This base set was extended to include features based on a small history window of previous sentences and "difference" features based on the query sentences. All of these were then used to build a decision tree using C4.5 [11] to predict non-relevant sentences. The intent was to find the most highly discriminative features, and hopefully increase precision by removing sentences from the candidate list which were likely to non-relevant based on the decision of the classifier.

Proximity Rule

This method is a simple form of clustering, where we assume that most relevant sentences occur in close proximity to "highly" relevant sentences. Based on our examination of the data, we found that a high proportion of sentences with a high tf.idf score were relevant, but there was very little overall correlation between tf.idf score and relevancy. On the other hand, there was a high correlation between a sentence's relative distance to a highly relevant sentence and its relevancy. This rule uses the tf.idf results, but unlike the other two methods is not restricted to that list when looking for sentences. We scan all the sentences in a document, using the tf.idf scores and relative distances as input. The proximity model can indicate that a sentence is relevant even if it received a zero score in the tf.idf model.

Here is one example of a simple proximity model which calculates a relevance score $R(i)$ for the i-th sentence in a document. It uses a window of nearby sentences with indexes $\{i - L, \ldots, i + K\}$, and scores $\{S(i - L), \ldots , S(i + K)\}$, where $S(i)$ is the tf.idf score for sentence i, and K and L are small positive integers (typically 2 or 3).

$$R(i) = f(S(i), i) \cdot \sum_{j=i-L}^{i+K} g_j(S(i), S(j), i, j)$$

Here, the function f is used to weight the contribution of sentence i based on its score and, optionally, its position in the document. The function g_j models the pair-wise relationship between sentences within the window, based on their scores and relative distances.

4.2 Finding Redundant Sentences

In this evaluation we focused on comparing *pairs* of sentences rather than more general *sets* of sentences. There are two main reasons for this. First, dealing with pairs is simpler and gives a good starting point before looking at the more general case. Second, when asked to find redundant facts in lists of sentences, most human assessors [20] (and we suppose, users) tend to focus on sentence pairs instead of complex subsets.

We view one redundant sentence as being a statistical translation of another. If we can build a good translation model in the language then we should be able to detect when two sentences are translations of the same thing. The task is simplified a little by the fact that the source and target sentences are in the same language. The methods we adopted for TREC use WordNet to estimate similarity for words and short phrases, and shallow parsing to help extract and compare sentence structures.

Given two sentences to compare, the algorithm has the following stages. First, we obtain parse trees for each sentence. For the TREC evaluation these were all pre-computed since parsing can be a time-consuming process. Second, we convert each parse tree into a graph that describes the modification structure of the terms. Third, we perform a simple graph matching algorithm that compares the terms from each sentence, weighted by their possible importance. The end result is a similarity measure that estimates how much one sentence is a translation of the other in the same language.

4.3 A Statistical Method for Estimating Word Semantic Similarity

Before looking at sentence structure, we estimate a similarity score for each word or short phrase. We do this by comparing the contexts in which they occur. We use a specialized set of contexts: those derived from each of the basic relation types available for a word or phrase in WordNet.

Given two words or short phrases to compare, we first construct a unigram model of the context for each word. Each unigram model is a linear combination of unigram sub-models. There is one sub-model for each relation we used from WordNet, which were synonyms, hyponyms, hypernyms, and coordinate terms. For some words, some of these submodels will be empty if no relation exists. Each submodel is built from the terms appearing in the word lists and, optionally, the glossary entries for that relation. A set of mixture weights is used to combine the probabilities from the submodels to calculate the final unigram model. The weights are trained from a training set of redundant, semi-redundant, and unrelated sentences.

If we visualize the enormous graph of words that comprise WordNet, each word has a set of synonyms and other related words. These together form a subgraph associated with that word. To compare two words we are essentially measuring the weighted overlap between their two subgraphs.

We compute the distributional similarity of the two overall unigram models using skew divergence. Skew divergence is described by Lee [6] and has the advantage of competitive predictive performance on statistical language tasks similar to ours, without requiring sophisticated smoothing. If $D(r \,\|\, q)$ represents the KL divergence between distributions r and q, the skew divergence is:

$$s_\alpha(q, r) = D(r \,\|\, \alpha q + (1 - \alpha) r)$$

where α is a smoothing parameter and is set to α=0.99 in our application.

Once the skew divergence is calculated, the score must be normalized. This is done by calculating a similarity score relative to a small fixed set of "unfamiliar" words that are extremely unlikely to all be similar to the target word. The final score is a real number between zero (identical match) and an arbitrary upper bound of 500 (maximum dissimilarity). Table 8 shows scores for the word "astronaut" compared to various words. Note that words like "orbit" share similar co-occurrence distributions with "astronaut" but, correctly, do *not* get low translation distance scores.

astronaut	astronaut	0.000
astronaut	cosmonaut	0.002
astronaut	man	9.051
astronaut	explorer	14.372
astronaut	commander	20.877
astronaut	pilot	33.503
astronaut	traveler	49.312
astronaut	watermelon	153.548
astronaut	orbit	283.162
astronaut	rocket	294.722
astronaut	committee	302.601

Table 8: Semantic similarity "distances" of various words from the word "astronaut", as measured by normalized skew divergence of Wordnet-based unigram mixture models.

Using distributional similarity may be seen as a type of query expansion. Unlike typical scenarios for query expansion, the terms being compared are coming from documents already deemed relevant, so the same word found in two different sentences is less likely to be used with two very different senses, making sense disambiguation less of a problem. There are other methods described to estimate the substitutability of words, e.g. the confusion probability [2]. We do not explore those here; Lee does a comparison of some of these in a recent paper [5].

4.4 Sentence Parsing and Modifier Graphs

After obtaining word translation probabilities, we analyze sentence structure by obtaining a parse of all sentences to be examined either in the document set or in the TREC description and narrative fields. There are currently two purposes for this parse. First, we pre-process surface features to be used by the relevancy classifier, and second, we derive a dependency graph from the parse tree to estimate headwords and modifier relations, and the relative "importance" of terms in a sentence, both of are used our simple translation model. For the actual parsing we used the Apple Pie Parser, an easy-to-use corpus-based probabilistic parser written in C and developed by S. Sekine [13].

The algorithm for converting a parse tree to a dependency graph can be defined recursively, starting at the leaves of the tree:

1. A terminal node (leaf) depends only on itself and has itself as a headword.
2. Each possible type of non-terminal node has a rule to decide on the headword for that constituent, given the headwords derived from its leaves. The "winning" headword then becomes a new node in the dependency graph, with the "losing" headwords pointing to the new node and thus becoming dependants of the new node.
   For example, for the noun phrase $NP \rightarrow n_1\ n_2\ n_3$, we will choose n_3 (the last noun) as the headword, creating a new node for it, and creating edges pointing from n_1 and n_2 to n_3.
3. This process is continued up the tree until the root is reached.

With this dependency graph and the word similarity scores, we can compute the final translation probability of two sentences.

4.5 Graph Matching

We use a very simple graph matching step to model the fact that not all words in a sentence are equally "central". Some words or short phrases express the core ideas in a sentence, and other words act to modify them. Therefore, it seems reasonable to weight any matches between the core ideas in two sentences more highly than matches between other words. We currently define the graph weight of a node as its in-degree.

Our graph matching is currently "greedy": for each word W_i with graph weight A_i in the source sentence, we select the word $V(i)$ with graph weight $B_{V(i)}$ with lowest similarity distance $S_{i,V(i)}$ in the target sentence. This is constrained by a limit of at most K matches to any target word, where $K = 2$ in our implementation. Once a target word has reached its match limit, the word with the next-lowest distance is used instead. The weighting factor for the i-th observation is $A_i \cdot B_i$, and so the matching score between the sentences is:

$$M(A,B) = \sum_{i=0}^{N} A_i B_{V(i)} S_{i,V(i)}$$

where N is an adjustable parameter (typically between 6 and 10) to be used when the source words are sorted in descending order of influence.

We show an example below for two sentences A and B taken from the TREC sample documents. The similarity score of Sentence A against Sentence B is the weighted mean described above: 14.821. Since this is below our threshold of 15, these sentences would be considered redundant.

Sentence A:

> Some of the best shots, **released** this month by the US space agency **Nasa**, **show** parts of the universe billions of light years away - and therefore **billions** of **years** in the **past**.

Sentence B:

> The images sent back this **year**, after astronauts repaired the telescope's defective mirror, **show** a **myriad** of astronomical objects too distant to be seen with the most **powerful** Earth-bound observatories.

Sentence A	Graph weight A_i	Sentence B (Most similar word)	Graph weight B_i	Similarity Distance S_i
past	6	year	4	2.456
years	3	year	4	0.0258
released	4	show	2	53.631
Nasa	5	powerful	1	68.240
show	2	show	2	0.000
billions	3	myriad	1	0.152
Weighted mean: 14.821				

Table 9: Comparison of word pairs from Sentence A and B, in order of influence. Only the top six word pairs, as sorted by Sentence A graph weight, are used for this example. Each word from Sentence A is paired with the word from Sentence B with the lowest similarity distance.

There is still plenty of work to do on the best features to represent in the graph, and the most appropriate, theoretically justified matching algorithm.

4.6 Novelty Track Results

We now give a brief summary of our official results, for both relevance and novelty, for the five runs we submitted. The best-performing runs are shown in boldface.

The runs are labeled as 'relevance algorithm + novelty algorithm'. The relevance algorithms Simple_Threshold, Decision_Tree (DT), and Proximity are those described previously. The novelty algorithms LowSameDoc and HighDiffDoc use the simple statistical translation approach. These labels refer to, respectively, a low threshold (sentences must be very similar in meaning) comparing sentences only within the same document, or a high threshold (more relaxed redundancy definition) comparing sentences only across different documents.

The official TREC scores we achieved for each run are shown in Table 10. which gives average precision (Ave P), average recall (Ave R), and average P·R scores. Our best average P·R score for relevance (0.058) was achieved with a simple tf.idf threshold on the ranked list of sentences. Our best average P·R novelty score (0.047) was achieved by selecting highly relevant sentences with the Proximity rule and then accepting all such candidates as novel, with our statistical approach performing marginally worse when applied to all sentences within the same document.

Run	Relevance			Novelty		
	Ave P	Ave R	Ave P·R	Ave P	Ave R	Ave P·R
Proximity + LowSameDoc	0.13	0.31	0.052	0.12	0.30	0.046
Proximity + HighDiffDoc	0.13	0.31	0.052	0.12	0.16	0.025
Proximity + All Novel	0.13	0.31	0.052	0.12	0.31	**0.047**
DT + LowSameDoc	0.10	0.13	0.019	0.10	0.13	0.018
Simple_Threshold + HighAllDoc	0.17	0.23	**0.058**	0.16	0.18	0.043

Table 10: Official Novelty Track Results by Run.

Comparative results, relative to the median P·R across all systems, are given in Table 11. The results are given as a fraction of the total number of queries (49). Because the scoring scale is continuous, we label as "at the median" any P·R score within ±0.01 of the median P·R score. Three of the runs used the same relevance algorithm (Proximity) and so these are collapsed into one entry. Overall, 4 out of 5 of our novelty runs had more than 50% of the scores at or above the median. Our best run, Proximity + All Novel, had 42.9% of scores above the median, and 91.8% of scores at or above the median.

Run (relevance + novelty algorithm shown)	Below Median	At Median	Above Median	At or Above Median
Relevance				
Simple_Threshold	0.367	0.347	**0.286**	0.633
Decision_Tree (DT)	0.694	0.265	0.041	0.306
Proximity	0.204	0.510	**0.286**	**0.796**
Novelty				
Simple_Threshold + HighAllDoc	0.286	0.388	0.327	0.714
Proximity + LowSameDoc	0.796	0.204	0	0.204
Proximity + HighDiffDoc	0.347	0.490	0.163	0.653
Proximity + All Novel	0.082	**0.490**	**0.429**	**0.918**
DT + LowSameDoc	0.469	0.408	0.122	0.531

Table 11: Comparative Novelty Track Results for Ave P·R scores, as a fraction of the total number of queries Runs labeled "at median" have an average P·R score within 0.01 of the median.

4.7 Conclusions

The problem of finding *specific* relevant sentences seems quite difficult. Easier and perhaps more helpful would be to find precise *zones* of relevance which include more context. In any case, our simple proximity model gave better performance than a more sophisticated decision tree method. The decision tree did not include the same proximity model, so there is some chance that combining the two methods might give better performance: the representation may have made the difference here. The very high proportion of sentences judged as novel made it easy for the trivial "accept everything as novel" algorithm to do well. As a result, overall system performance on this track was dominated by the ability to find relevant sentences.

We described a word semantic similarity measure based on comparing word contexts from WordNet. Other query-expansion-type techniques, such as LSI might work as well or better. Wordnet is interesting because it allows some flexibility in how different similarity "features", such as synonyms, hyponyms, coordinate terms, and so on, are combined. Unfortunately, calling Wordnet and building language models is very slow, so pre-computing the LSI matrix might be a good compromise.

The greedy graph-matching approach is a first step in a direction we think is promising. It's clear that using only the in-degree of word nodes is not a sufficient indication of their importance in many cases. The current algorithm does tend to find good, similar sentences, but is still too tolerant of differences in the lesser-weighted areas of the graph. Among other things, named entities could use special treatment. With more work we think it should be possible to create a much more accurate alignment model for redundant sentences and passages.

5. Acknowledgements

This material is based on work supported by NSF grants IIS-0096139 and EIA-9983253. Any opinions, findings, conclusions, or recommendations expressed in this material are the authors', and do not necessarily reflect those of the sponsor.

6. References

[1] A. Arampatzis, J. Beney, C.H.A. Koster, and T.P. van der Weide. KUN on the TREC-9 Filtering Track: Incrementality, decay, and threshold optimization for adaptive filtering systems. In *Proceedings of Ninth Text REtrieval Conference (TREC-9)*, 2001.

[2] R. Grishman and John Sterling. Generalizing automatically generated selectional patterns. In the *Proceedings of the 15th. International Conference on Computational Linguistics*, Kyoto, Japan. Vol. II, 1994, pages 742-747.

[3] D. Hawking and N. Craswell. Overview of the TREC-2001 Web Track. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, 2002, pages 61-67.

[4] W. Kraaij, T. Westerveld, and D. Hiemstra. The Importance of Prior Probabilities for Entry Page Search. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*, pages 27-34.

[5] L. Lee. On the Effectiveness of the Skew Divergence for Statistical Language Analysis. In *Artificial Intelligence and Statistics 2001*, pages 65-72.

[6] L. Lee. Measures of Distributional Similarity. In the *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 1999, pages 25-32.

[7] The Lemur toolkit for language modeling in information retrieval. http://www.cs.cmu.edu/~lemur

[8] T. Minka. Bayesian Analysis of a Threshold Classifier, 2001. http://www.stat.cmu.edu/~minka/papers/minka-threshold.ps.gz

[9] P. Ogilvie and J. Callan. Experiments Using the Lemur Toolkit. In *Proceedings of the Tenth Text Retrieval Conference, (TREC-10)*, 2002, pages 103-108.

[10] J. Ponte and W. B. Croft. A Language Modeling Approach to Information Retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1998)*, pages 275-281.

[11] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[12] S.E. Robertson. Threshold setting in adaptive filtering. In *Journal of Documentation*. In press.

[13] S. Sekine. The Apple Pie Parser. http://www.cs.nyu.edu/cs/projects/proteus/app/

[14] T. Westerveld, W. Kraaij, and D. Hiemstra. Retrieving Web Pages Using Content, Links, URLs, and Anchors. In the *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, 2002, pages 663-672.

[15] J. Xu and W. Croft. Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd Annual International Conference ACM SIGIR on Research and Development in Information Retrieval (SIGIR 1999)*, pages 254-261.

[16] C. Zhai and J. Lafferty. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001)*, pages 334-342.

[17] Y. Zhang, and J. Callan. YFilter at TREC9. In *Proceeding of Ninth Text REtrieval Conference (TREC-9)*, 2001

[18] Y. Zhang, and J. Callan. YFilter at TREC10. In *Proceeding of Tenth Text REtrieval Conference (TREC-10)*, 2002.

[19] Y. Zhang, J. Callan. Maximum Likelihood Estimation for Filtering Thresholds. In *Proccedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001).*

[20] Y. Zhang, J. Callan, and T. Minka. Novelty and Redundancy Detection in Adaptive Filtering. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002)*, pages 81-88.

[21] Y. Zhang, W. Xu, J. Callan. Exact Maximum Likelihood Estimation for Word Mixtures. In *Text Learning Workshop in International Conference on Machine Learning (ICML2002).*

Video Classification and Retrieval with the Informedia Digital Video Library System

A. Hauptmann, R. Yan, Y. Qi, R. Jin, M. Christel, M. Derthick,
M.-Y. Chen, R. Baron, W.-H. Lin, and T. D. Ng.

Carnegie Mellon University,
School of Computer Science,
Pittsburgh PA, 15213-3891 USA

This paper is organized in three parts. The first part details some of the lower level shot classification work, the second part describes the 'manual' retrieval systems while the last section details the interactive retrieval system for the Carnegie Mellon University TREC Video Retrieval Track runs. The description of the data can be found elsewhere in the proceedings of the 2002 TREC conference video track overview.

Classification

In the TREC02 video track, one of the main tasks is to detect various semantic features concepts such as "Indoor/Outdoor", "People" etc. This part contains the description of the classification tasks. We submitted runs for the following classification concepts in the TREC 2002 Video Track. To obtain training data, we manually annotated each I-frame of the 23.26 hours feature development collection for each category.

a. **Outdoors**: Segment contains a recognizably outdoor location, i.e., one outside of buildings. Should exclude all scenes that are indoors or are close-ups of objects (even if the objects are outdoor).
b. **Indoors**: Segment contains a recognizably indoor location, i.e., inside a building. Should exclude all scenes that are outdoors or are close-ups of objects (even if the objects are indoor
c. **Cityscape**: Segment contains a recognizably city/urban/suburban setting.
d. **Monologue:** an event in which a single person is at least partially visible and speaks for a long time without interruption by another speaker
e. **Face:** at least one human face with nose, mouth, and both eyes
f. **People:** a group of two more humans
g. **Text Detection:** superimposed text large enough to be read
h. **Speech:** human voice uttering recognizable words
i. **Instrumental Sound:** sound produced by one or more musical instruments. including percussion instruments

Feature Extraction

It is a critical challenge to find a good feature set for image classification. A number of image features based on color and texture attributes have been reported in the literature for image retrieval. We tried several of them and explored some new features at the same time.

Color Histograms. We used the histogram of 3*3 image regions in HSV color space for each MPEG I-frame. The color features were derived from a histogram in the quantized HSV color space.

Textures. We use the mean and variance of a texture orientation histogram for each of the 3*3 regions as texture feature.

Edge features. We used a feature called the Edge Direction Histogram. A Canny edge detector was used to extract the edges from an image. A total of 73 bins were used to represent the edge direction histogram of an image; the first 72 bins are used to represent the edge directions quantized at 5^{o} intervals and the last bin represents a count of the number of pixels that didn't contribute to any edge.

Edge direction coherence vector. This feature stores the number of coherent versus non-coherent edge pixels with the same edge directions (considering only horizontal and vertical axis within a range of +/- 5^{o},). We thresholded on the size of every 8 connected components of edges in a given direction to decide whether the region could be considered coherent or not. This feature was used to distinguished structured edges (like edges of buildings) from arbitrary edge distributions.

Camera motion. We used statistical distribution patterns to detect the pan/tilt/zoom camera operations based on the motion vectors of MPEG encoding. The resulting features encoded the presence/absence of these six kinds of camera operations (pan left, pan right, tilt up, tilt down, zoom in, zoom out) as a new type of feature for image classification.
MPEG motion vectors. We transformed the motion vectors directly encoded in the MPEG-compressed video into a different kind of feature, namely a histogram of the motion vector angle and velocity, as well as the wavelet coefficients of motion vectors..
Although we experimented extensively with the features derived from camera motion analysis and the raw MPEG motion vectors, these additional features did not contribute to overall classification accuracy.

Classification Algorithms

We experimented with several classification tools for these tasks, including SVM, KNN, Adaboosting and Decision Trees. Comparing their performance using cross validation on a comparative large data set, we reached the conclusion that support vector machine learning was best, with the power=2 polynomial as the kernel function. Nonlinear functions usually performed better than linear SVM kernel functions. The trade-off is that for nonlinear functions, the parameter space can be huge and therefore it may cause overfitting for small training datasets.
Among the tasks, the cityscape classification suffered from the problem of insufficient positive training examples, which is also the reason why we did not submit a landscape classification for evaluation. For the cityscape classification training data, the positive examples (that is, the cityscape images vs. the non-cityscape images) comprised only 12% of the whole data set. Such small ratios of positive examples in the training set cannot be well represented by the classification methods we attempted. In addition, we investigated using the chi-square function as distance function based on published literature. Contrary to published claims, the chi-square function was not superior to any other functions.

Cross_validation
Due to the temporal correlation between adjacent images in a video, an initial cross validation based on random sampling of shots gave much better performance than appropriate for the true prediction capability of the models. This was due to the fact that similar shots appeared throughout a single video or 'movie'. So we performed a video based cross validation based, using 30 complete videos as training and then testing on the remaining 11 videos.

Feature Selection
It is a challenge to select a good feature set for image classification. Qualifying their discrimination ability of each feature in the given classification problem is difficult. We performed video-based cross validation on training sets and compared the different features' performance based on the resulting classification error and precision / recall of each task.
For the camera motion related features and the MPEG motion vector related features, we explored numerous experiments to test their usefulness to the image classification task. However, they did not give conclusive results clearly. Finally, we ended up not using the camera or motion features in the final submission.

To get the best feature combination for each task, we performed a 6 folder movie based cross validation on the three training sets on different feature combinations. The best feature combinations were always included texture, edge and color features . Since the results were submitted as shot based features and not classification from individual images, we integrated all I frame classification results in a shot into this shot's feature detection result. The confidence of a particular feature detection is the ratio between number of feature presenting I frames vs. number of feature absent I frames in this shot.
Our results showed huge difference of the performance of different classifiers. The reason of this discrepancy is possibly caused by the variability of the training sets, the inconsistency between training set and the test sets, or the varying difficulty of the different classifications.

Non-standard classification for text, faces, people, monologues, speech and music

Variations of the classification approach were used for the face, people. monologue and audio categories.
For the **face** category, we used the Schneiderman face detector [19], exclusively.
For **people**. we extracted the following features
At the level of shots:

- Number of frames in a shot

- Number of faces detected by the face detector
- Number of faces with high confidence
- Number of faces with low confidence
- Average confidence score of the faces in a shot,
- The standard deviation of the face scores,
- A smoothed minimum face score,
- A smoothed maximum score,
- Average pixel area for each detected face.

For each I-frame within a shot we also extracted these frame-based features:

- Average number of faces per frame,
- Average number of faces per frame with high confidence
- Average number of faces per frame with low confidence

Since the total number of features was fairly low, we trained a decision-tree based classifier (C4.5), which outperformed SVM on this task in cross-validation experiments.
Our contrasting **people** classification submission merely counted the number of faces visible in each I-frame, and averaged this over the whole shot. This baseline approach performed significantly worse with a classification error of 0.403 vs. 0.498.

The task of **text-overlay** classifier is to find scenes with superimposed texts. Simply predicting a scene to be a text overlay based on whether or not the OCR engine is able to find text is not good enough because that OCR engine is quite error-prone. The features extracted were:

1. time: related to the whole movie, when is the OCR detected texts are found
2. #terms_within_a_shot
3. #dictionary_words_within_a_shot
4. average_popularity_valid_trigram_in_a_shot
5. average_popularity_valid_4gram_in_a_shot
6. average_no_alphabets_found_in_a_term
7. ratio_dictionary_words_to_detected_terms
8. ratio_length_of_all_dictionary_words_to_length_of_detected_terms

For classification, similar to the people classifier, a decision tree (C4.5) was used instead of a SVM.

For **monologues**, we used as features:
1. The portion of time where a least one (face) was detected.
2. The confidence of the face in every I-frame.
3. The number of speaker voice changes in one shot
4. The confidence in any significant audio change during this shot.
5. The number of faces present in one image.
These features were also fed into an decision tree classifier.

Speech and music were classified by the same speaker identification code as in the 2001 TREC video track.

'Manual' video retrieval with classification pseudo-relevance feedback

Example-based image retrieval task has been studied for many years. The task requires the image search engine to find the set of images from a given image collection that is similar to the given query image. Traditional methods for content-based image retrieval are based on a vector model. These methods represent an image as a set of features and the difference between two images is measured through a (usually Euclidean) distance between their feature vectors. While there have been no large-scale, standardized evaluations of image retrieval systems, most image retrieval systems are based on features such as color, texture, and shape that are extracted from the image pixels.

In our system two kinds of low-level features are used for finding similar images: color features and texture features. The color features are the cumulative color histograms for each separate color channel, where the three channels are derived from the HSV color space. We use 16 bins for hue and 6 bins for both saturation and value. We generate a texture feature for each subblock of a 3*3 image tessellation. The texture features are obtained through the convolution of the subblock with various Gabor Filters. In our implementation, 6 angles are used and each filter output is quantized into 16 bins. We compute a histogram for each filter and generate their central and second-order moments as the texture features. We concatenate all the features into a longer feature vector for every image; i.e. one vector for all color features and one vector for all texture features. We use a simple nearest neighbor (NN) image matching algorithm on both color and texture to produce the initial similarity results. In a preprocessing step, each element of the feature vectors is scaled by the covariance of its dimension. We adopted the Euclidean distance as the similarity measure between two images.

Although nearest neighbor search is the most straightforward approach to finding the matching images, it suffers from two major drawbacks. First, irrelevant features in the vector are given equal weight to important features, and thus retrieval accuracy will hurt decrease dramatically. Feature selection is therefore a necessary step prior to computing the nearest neighbor images. In theory, relevance feedback, through re-weighting and query refinement, is a powerful tool to refine the feature weighting so as to provide more accurate results. However, it is impossible to obtain the user judgment information in most automatic retrieval tasks. A second negative aspect is the unjustified distance function. Since an appropriate distance measure is a function of both the characteristics of the dataset and of the queries, a simple Euclidean distance function is unlikely to work for all the queries and images. Another concern is the normalization of the different dimension of a feature vector. To mitigate all these issues, we propose a classification-based pseudo-relevance feedback approach to refine the initial retrieval result. Support Vector Machines (SVMs) are used as our basic classifier mechanism, since SVMs are known to yield good generalization performance compared to other classification algorithms.

.The basic idea for this approach is to augment the retrieval results by incorporating the classification output value through Pseudo-Relevance Feedback (PRF). The input data for the classifier is based on the information provided by our initial retrieval results. Standard PRF methods, which originated in the text information retrieval community, utilize the top-ranked documents as positive examples to improve the accuracy. The idea is to re-weight the words in the document feature vector based on the words in the top ranked documents, which are assumed to be positive examples. However, due to the poor initial performance of current video retrieval system, even the very top-ranked results are not always the correct ones that meet the users' information need. Unlike in text retrieval methods, it is more appropriate to make use of the *lowest* ranked documents in the collection after the initial search, which are more likely to be the negative examples. Therefore, we construct a classifier where the positive data are the query image examples and the negative data are sampled from the least confident image examples in the initial retrieval results.

Since the number of positive examples in our retrieval task is always much smaller than the number of the negative examples, we cast the problem into the imbalanced dataset classification framework. To sample more negative examples but achieve an overall balanced distribution of negative and positive examples in the classifier training set, we apply an ensemble of SVMs to tackle the rare class problem. The overall procedure can be summarized as follows,

1. Generate the initial classification results by nearest neighbor retrieval for all the images in the collection.
2. Choose all the query images as positive data. Denote the number of query images as *m*.
3. Construct a negative sub-collection based on the initial retrieval results, which are defined by the lowest 10% of the retrieved data from the collection. We sample *k* groups of negative data from the negative sub-collection, where each group contains *m* query images. Combine each group of negative data and all the positive data as a training set.
4. Build a classifier from each training set to produce new relevant score for any images *x* $f_i(x)(1 \le i \le k)$. where *i* is the index of training set
5. Combine the results in form of logistic regression, which is

$$P(+|x) = \frac{\exp(\beta_0 + \sum_{i=1}^{k} \beta_i f_i(x))}{1 + \exp(\beta_0 + \sum_{i=1}^{k} \beta_i f_i(x))}$$

In our system, we simply set β_0 as 0, $\beta_i(1 \le i \le k)$ as equal values.

Our approach presented here utilizes the collection distribution knowledge to refine the final result. Due to the good generalization ability of the SVM algorithm, the most relevant features are selected automatically. Also the approach yields a better distance function based on the probability estimation compared with the simple Euclidean distance.

Combination of multiple agents

As the first step to integrate different types of agents, all the relevance scores of the agents are converted into posterior probability. For each agent other than the classification-based PRF agent, the posterior probability is generated by a linear transformation of their rank and scaled to the range of [0, 1]. All these posterior probabilities are simply linear combinations as follows:

$$Score = a_I(b_c P_{color}(+\mid x) + b_t P_{texture}(+\mid x) +$$
$$b_{PRF} P_{PRF}(+\mid x)) + a_T P_{text}(+\mid x) + a_m P_{movie}(+\mid x)$$

where a_I, a_T, a_m is the weight for image agent, text agent, movie information agent respectively, which are set to be 1, 1, 0.2. b_c, b_t, b_{PRF} are the weights for the three search agents for image retrieval: NN on color, NN on texture and classification PRF, which are either set to be 0 or 1 in our contrastive experiments reported below.

Speech Recognition

The audio processing component of our video retrieval system splits the audio track from the MPEG-1 encoded video file, and decodes the audio and downsamples it to 16kHz, 16bit samples. These samples are then passed to a speech recognizer. The speech recognition system we used for these experiments is a state-of-the-art large vocabulary, speaker independent speech recognizer. For the purposes of this evaluation, a 64000-word language model derived from a large corpus of broadcast news transcripts was used. Previous experiments had shown the word error rate on this type of mixed documentary-style data with frequent overlap of music and speech to be 35 – 40%.

Text Retrieval

All retrieval of textual material was done using the OKAPI formula. The exact formula for the Okapi method is shown in Equation (1)

$$Sim(Q,D) = \sum_{qw \in Q} \left\{ \frac{tf(qw,D)\log(\frac{N - df(qw) + 0.5}{df(qw) + 0.5})}{0.5 + 1.5\frac{|D|}{avg\_dl} + tf(qw,D)} \right\} \qquad (1)$$

where *tf(qw,D)* is the term frequency of word *qw* in document *D*, *df(qw)* is the document frequency for the word *qw* and *avg_dl* is the average document length for all the documents in the collection.

Results

We report our results in terms of mean average precision in this section, as shown in Table 1. Four different combination of the retrieval agents are compared in this table, including the combination of text agents (Text), movie agents (Movie), nearest neighbor on color (Color), nearest neighbor on texture (Texture) and classification-based PRF (Classification). The results show a significant increase in retrieval quality using classification-base PRF technique. While the text information from the speech transcript accounts for the largest proportion of the mean average precision (0.0658), only a minimal gain was observed in the mean average precision when the 'movie title' and abstract were also searched (0.0724) in addition to the speech transcripts. The image retrieval component provided further improvements in the scores to a mean average precision of 0.1046. Finally, the PRF technique managed to boost the mean average precision to the final mean average precision score of 0.1124.

Approach	Precision	Recall	Mean Average Precision
Text only (ASR)	0.0348	0.1445	0.0658
Text + Movie information (Abstract and Title)	0.0348	0.1445	0.0724
Text + Movie + Image retrieval (Color + Texture)	0.0892	0.220	0.1046
Text + Movie + Color + Texture + PRF Classification	0.0924	0.216	0.1124

Table 1 Video Retrieval Results on the 25 queries of the 2003 TREC video track evaluation.

Interactive Video Retrieval

For the 2002 TREC video track interactive condition, we used the basic Informedia Digital Video Library system, as in the 2001 TREC Video TREC. A few refinements to the interface are discussed and illustrated below.

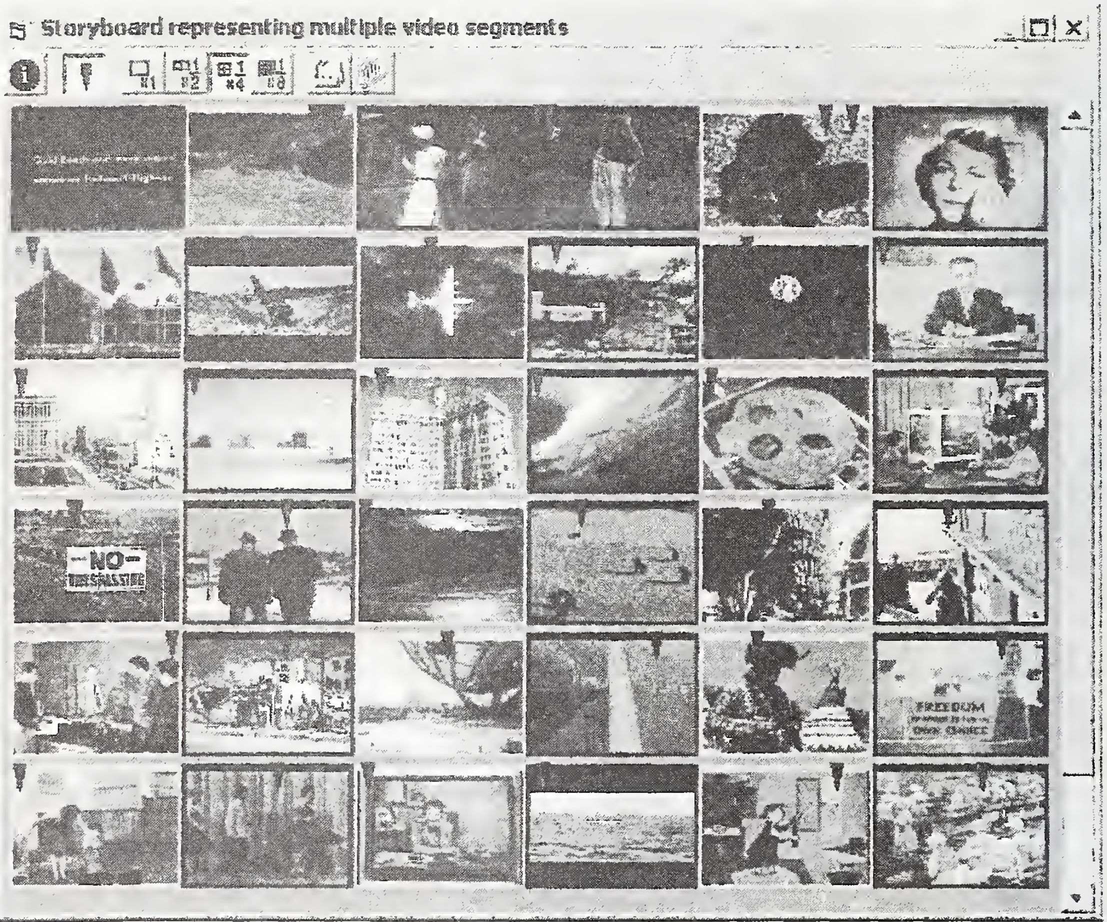


Figure 1. Multi-document storyboards combine all shots from highly relevant segments into one display.

Since IDVLS was designed to return 'stories', which can encompass multiple shots as retrieval results. we modified the interface to allow a shot-based presentation of the results which we called "*Multiple document storyboards*". The text was retrieved in roughly 3-minute story chunks, and all shots for that story were presented to the user. A storyboard display, which concatenated the top N relevant stories and their shots, was used [Figure]. Thus a user could visually scan for relevant images from a fairly large storyboard display of the top relevant stories and their shots. Selecting a shot as relevant placed this shot onto an answer set display, which could again be edited before final submission [Figure].

Because of the large number of shots on the result storyboard, we placed the resolution of the keyframe size and the layout under user control. Thus a user can shrink or enlarge the size of the keyframes displayed on the storyboard, depending on the desire to visually inspect the keyframes more closely, or to view the complete set. The size of the window, and the total number of results displayed could also be modified. We found that the query context plays a key role in filtering image sets to manageable sizes. The TREC 2002 image feature set offered filtering capabilities for the classified categories of indoor, outdoor, faces, people, etc. The user interface provided for a display of the classified feature values for every shot [Figure]. The user was also able to control the threshold values for each of the feature categories. This enabled the display to be more manageable by filtering out shots that were more likely to

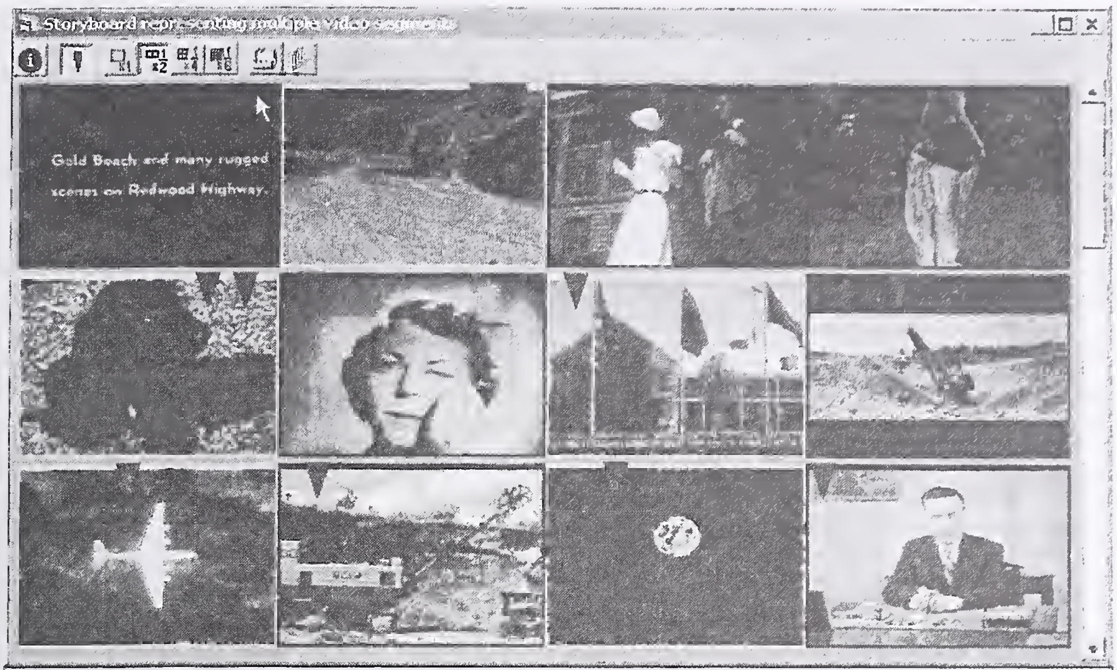
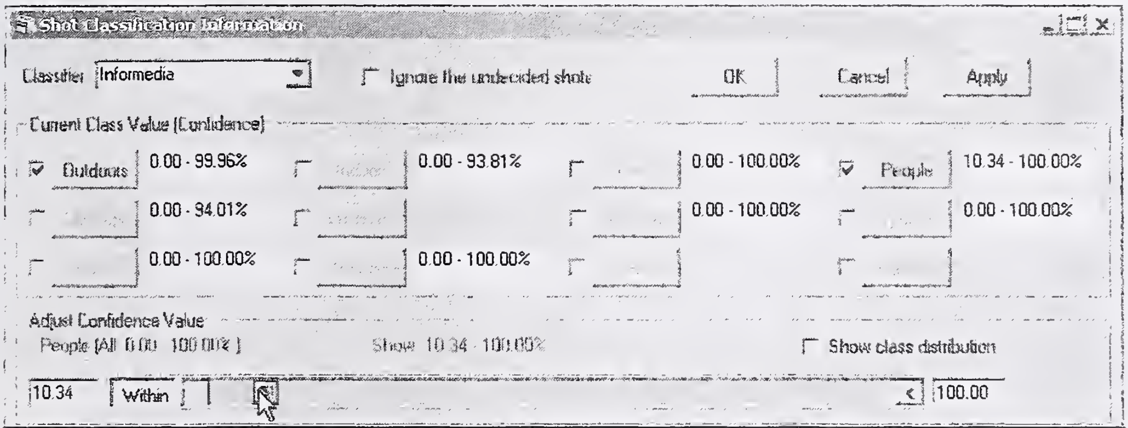

Storyboard representing multiple video segments
Gold Beach and many rugged scenes on Redwood Highway.

Figure 2. Resolution and layout of the storyboard can be modified by the user.

be feature X, and unlikely to be feature Y, depending on the query context. Since the display showed the number of active results, and provided direct feedback on the distribution of the data, the large number of irrelevant shot could easily be filtered down to a manageable number, that was then visually scanned by the user.

The multi-document storyboard facilitated quick inspection of many images. A first-order filtering by query text provided an initial set of images that constituted potential results. The multi-document storyboard based on 3-minute segments and shots enabled the user to find relevant shots, which were temporally near shots where query-words had been matched. The keyframe ordering by video segment and time useful. The classified shot features were useful for filtering, but needed to be manually adjusted depending on the particular queries. Users were able to drill-down to details, going from keyframe images to observing video, which was often necessary to eliminate uncertainty that could not be resolved by looking at a still image frame.



Classifier: Informedia

□ Ignore the undecided shots

Current Class Value (Confidence)

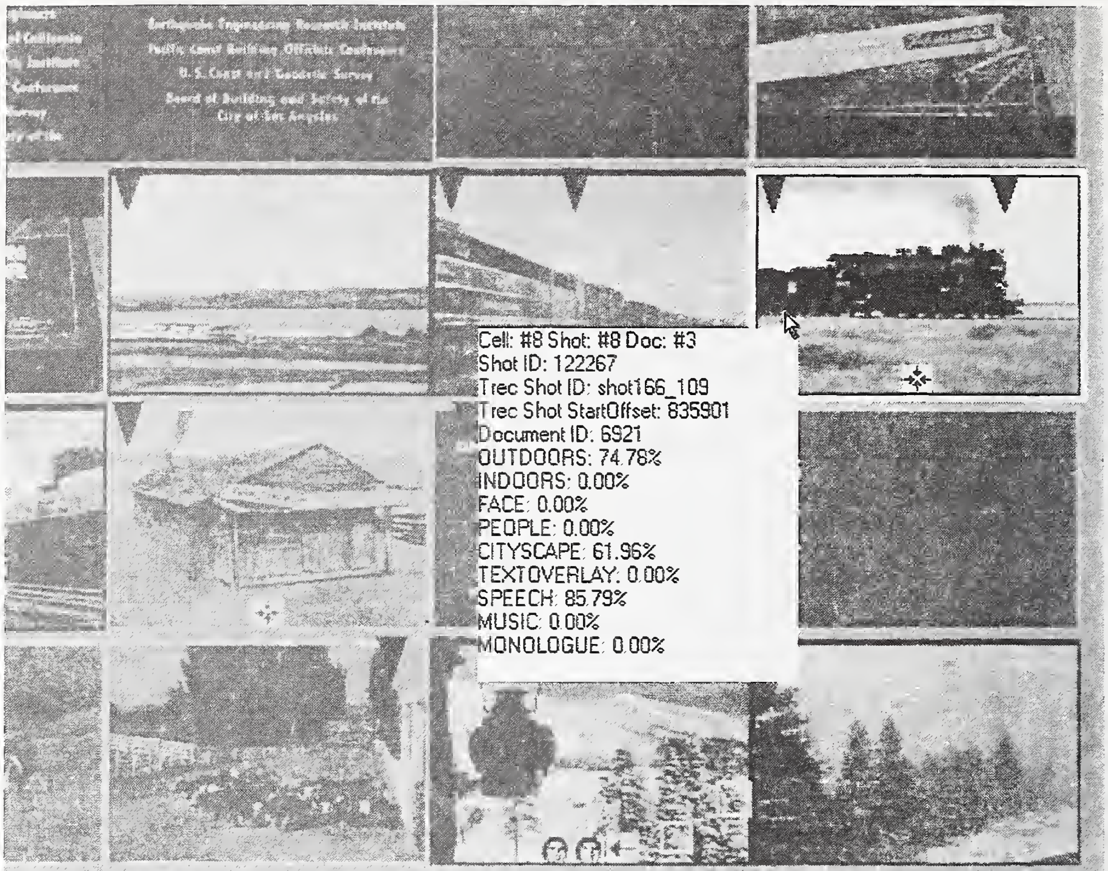
☑ Outdoors 0.00 - 99.96%	□ 0.00 - 93.81%	□ 0.00 - 100.00%	☑ People 10.34 - 100.00%
□ 0.00 - 94.01%	□	□ 0.00 - 100.00%	□ 0.00 - 100.00%
□ 0.00 - 100.00%	□ 0.00 - 100.00%	□	□

Adjust Confidence Value

People (All 0.00 100.00%) Show 10.34 - 100.00% □ Show class distribution

10.34 Within 100.00

Figure 3. Users can filter shots based on thresholds in any feature classification category.



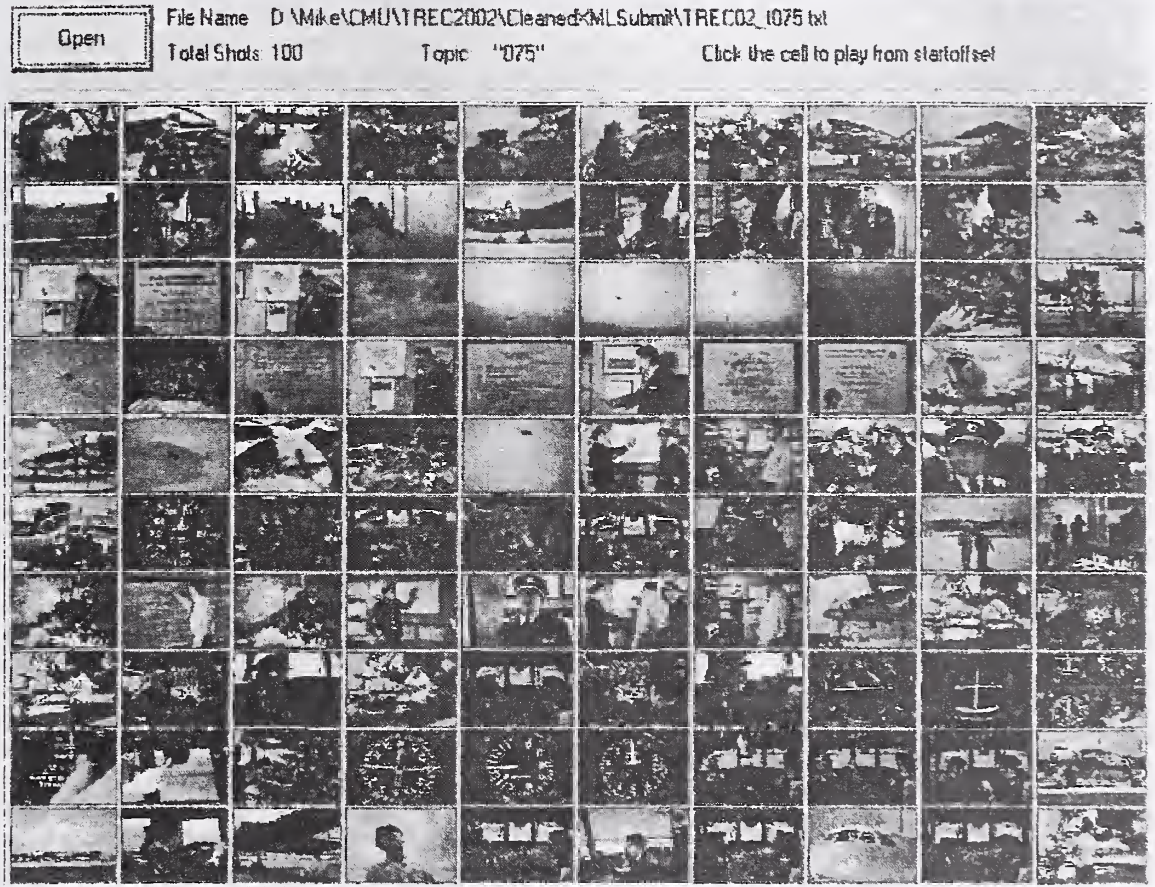
Cell: #8 Shot: #8 Doc: #3
Shot ID: 122267
Trec Shot ID: shot166_109
Trec Shot StartOffset: 835901
Document ID: 6921
OUTDOORS: 74.78%
INDOORS: 0.00%
FACE: 0.00%
PEOPLE: 0.00%
CITYSCAPE: 61.96%
TEXTOVERLAY: 0.00%
SPEECH: 85.79%
MUSIC: 0.00%
MONOLOGUE: 0.00%

Figure 4. Feature classification statistics are accessible for any shot.

References

1. Hafner, J. Sawhney, H.S. Equitz, W. Flickner, M. and Niblack, W. "Efficient Color Histogram Indexing for Quadratic Form Distance." IEEE Trans. Pattern Analysis and Machine Intelligence, 17(7), pp. 729-736, July, 1995.
2. Robertson S.E., et al., Okapi at TREC-4. In The Fourth Text Retrieval Conference (TREC-4). 1993.
3. Sato, T., Kanade, T., Hughes, E., and Smith, M. Video OCR for Digital News Archive. In *Proc. Workshop on Content-Based Access of Image and Video Databases.* (Los Alamitos, CA, Jan 1998), 52-60.
4. Singh, R., Seltzer, M.L., Raj, B., and Stern, R.M. "Speech in Noisy Environments: Robust Automatic Segmentation, Feature Extraction, and Hypothesis Combination," *IEEE Conference on Acoustics, Speech and Signal Processing*, Salt Lake City, UT, May, 2001.
5. A.W.M. Smeulders, M. Worring, S. Santini, A. Gupta and R. Jain, "Content-Based Image Retrieval at the End of the Early Years." IEEE Trans. Pattern Analysis and Machine Intelligence, 22(12), pp. 1349-1380, December. 2000.
6. Swain M.J. and Ballard, B.H. "Color Indexing." Int'l J. Computer Vision, vol. 7, no. 1, pp. 11-32, 1991.
7. Tague-Sutcliffe, J.M., "The Pragmatics of Information Retrieval Experimentation, revised." Information Processing and Management, 28, 467-490, 1992.
8. TREC 2002 National Institute of Standards and Technology, Text REtrieval Conference web page, http://www.trec.nist.gov/, 2002
9. The TREC Video Retrieval Track Home Page, http://www-nlpir.nist.gov/projects/trecvid/

10. Wactlar, H.D., Christel, M.G., Gong, Y., and Hauptmann, A.G. "Lessons Learned from the Creation and Deployment of a Terabyte Digital Video Library", *IEEE Computer* 32(2): 66-73.
11. *Informedia Digital Video Library Project Web Site*. Carnegie Mellon University, Pittsburgh, PA, USA. URL http://www.informedia.cs.cmu.edu
12. A. Del Bimbo " Visual Information Retrieval", Morgan Kaufmann Ed., San Francisco, USA, 1999
13. Mojsilovic, J. Kovacevic, J. Hu, R.J. Safranek, and S.K. Ganapathy, "Matching and Retrieval Based on the Vocabulary and Grammar of Color Patterns," IEEE Trans. Image Processing, 9(1), pp. 38-54, 2000
14. Gong, Y. *Intelligent Image Databases: Toward Advanced Image Retrieval*. Kluwer Academic Publishers: Hingham, MA.
15. A. Vailaya, A. Jain, and H.J. Zhang, "On image classification: city images vs. landscapes", *Pattern Recognition*, 31(12)(1998) 1921-1935.
16. Y. Li and L. G. Shapiro. "Consistent Line Clusters for Building Recognition in CBIR," International Conference on Pattern Recognition, August 2002
17. M. Szummer, R. W. Picard, **Indoor-Outdoor Image Classification**, IEEE International Workshop on Content-based Access of Image and Video Databases, in conjunction with ICCV'98
18. Q. Iqbal and J. K. Aggarwal, "Applying perceptual grouping to content-based image retrieval: Building images," in IEEE International Conference on Computer Vision and Pattern Recognition, Fort Collins, Colorado. vol. 1. pp. 42--48, June 1999.
19. H. Schneiderman, T. Kanade. "Probabilistic Modeling of Local Appearance and Spatial Relationships for Object Recognition." IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 45-51. 1998. Santa Barbara, CA.
20. A. G. Hauptmann, R. Jin, N. Papernick, D. Ng, Y. Qi, R. Houghton, and S. Thornton: Video Retrieval with the Informedia Digital Video Library System. The 10th Text REtrieval Conference (TREC 2001), National Institute of Standards and Technology (NIST) Gaithersburg, Maryland, 2001.



File Name D \Mike\CMU\TREC2002\CleanedXMLSubmit\TREC02_t075 txt

Total Shots 100 Topic "075" Click the cell to play from startoffset

Figure 5. The final result set can be reviewed and edited before submission.

The JAVELIN Question-Answering System at TREC 2002

E. Nyberg, T. Mitamura, J. Carbonell, J. Callan, K. Collins-Thompson,
K. Czuba, M. Duggan, L. Hiyakumoto, N. Hu, Y. Huang, J. Ko, L.V. Lita,
S. Murtagh, V. Pedro, D. Svoboda
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3891

1 Introduction

This paper describes the JAVELIN approach for open-domain question answering (Justification-based Answer Valuation through Language Interpretation), and our participation in the TREC 2002 question-answering track. The main scientific tenets underlying JAVELIN are:

- **QA as Planning**. Question Answering is a complex multi-faceted task, where question type, information availability, user needs, and NLP sophistication of QA modules all combine dynamically to determine the optimal QA strategy. JAVELIN includes modular infrastructure controlled by a planner, which combines analysis modules, information sources, user discourse and answer synthesis.
- **Universal Auditability**. Every step, from question analysis to answer generation, creates and updates a detailed set of labeled dependencies to form a traceable network of reasoning steps. This dependency network will provide the basis for refinement dialogs, reasoning maintenance, user-driven knowledge auditability, and machine learning for QA subtasks and control strategies.
- **Utility-based Information Selection and Fusion**. The expected utility of information is used to guide planning decisions and produce confidence scores for resulting answers, based on task context, information value to the user, resource constraints, and the accuracy of individual QA modules.

Past work on open-domain question answering has focused on particular aspects of the problem:

- *QA as Information Retrieval.* Systems that rely primarily on traditional IR techniques attempt to find the most relevant documents for a query. If shorter answers are required, some method is used to identify the most relevant passage in the document, typically via some passage ranking scheme based on query terms [12]. IR-based approaches can fail to answer a question when the answer is not directly found in the text, but must be inferred from the content.
- *QA as Information Extraction.* Systems that rely primarily on traditional IE techniques use a template-based approach widely explored by participants in the TIPSTER and MUC competitions. Questions and answers are associated with slot-filler structures, which are fully or partially matched by the contents of searched documents. Answers are provided by extracting the contents of filled templates. When used for QA, this approach can be linked with special forms of indexing, which tag named entities (people, organizations, locations, etc.) in the original corpus [15]. IE techniques alone have difficulty with broader, vaguer questions, where the answer must be derived by fusing facts from several passages, and cannot be located in one particular template fill.
- *QA as Natural Language Processing.* More recent systems have begun to improve on the performance of purely IR or IE based system through the use of NLP techniques that include deeper semantic analysis. A common use of NLP is in question analysis, where the original question is parsed to some degree, in order to understand it well enough to: a) perform selective query expansion based on meaning, rather

than a “bag of words”; b) focus the search for the answer on precisely the unit of meaning which is sought (e.g., the particular named entity that answers a “who is” or “where is” question) [4, 6, 11, 8]. To date, systems have focused on NLP and semantic processing as a way of “boosting” or filtering IR-based or IE-based methods. The limited scope, content and availability of large-scale ontological resources (such as WordNet) is a challenge.

The most successful recent systems, such as the Falcon system developed at SMU [8], acknowledge that a hybrid approach, with several levels of sophistication, can produce better results than systems that rely purely on IR or IE approaches with a bit of parsing thrown in. Perhaps the most efficient way to approach QA is to deploy the simplest technique that achieves a high-quality answer for a given question, on the assumption that simplicity implies a more rapid response. Significant progress on open-domain question answering could be achieved by systems that incorporate the most sophisticated language processing technologies, and fall back to simpler pattern-driven techniques where more sophisticated techniques fail. A general framework for multi-strategy QA will require advances in how QA systems are configured (using a modular architecture) and how they are controlled at run time (by incorporating explicit planning and reasoning components).

The remainder of this paper is organized as follows: Section 2 describes the JAVELIN system as implemented at the time of the TREC QA evaluation, and briefly presents several key extensions to the system completed shortly afterward; Section 3 presents results for the TREC QA task; Section 4 summarizes post-TREC analysis; and Section 5 closes with a discussion of lessons learned and future directions.

2 System Architecture

JAVELIN is based on a flexible, extensible, object-oriented architecture that separates the details of individual operations (e.g., taggers, parsers) from the context(s) in which they are used. The architecture is designed to support component-level evaluation, so that competing strategies and operators can be compared in terms of various performance criteria.

The current system, depicted in Figure 1, integrates a variety of modular components that perform individual question answering functions, including question analysis, document retrieval, answer candidate extraction, and answer selection. A centralized Planner module, supplied with a model of the question-answering task, selects and sequences execution of individual components, enabling the system to generate multiple processing strategies and replan when necessary. The actual execution details are handled by the Execution Manager, which also takes care of storing of session data (process steps, intermediate and final

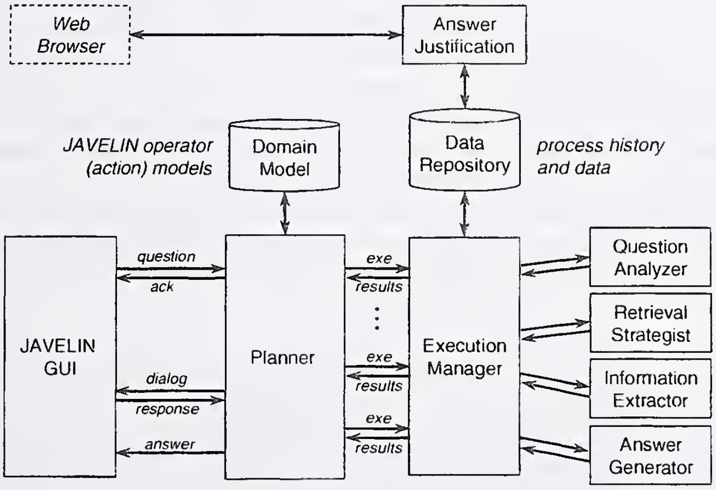


Figure 1: The JAVELIN architecture. The Planner operates as a service to the user interface, and controls execution of the individual components via the Execution Manager. Components included in the TREC test are shaded. Those shown in white were added subsequently.

results) in the Repository. User interaction is coordinated by the graphical user interface (GUI) and the Answer Justification module, which provides a browsable view of the process history.

It is important to note that only the shaded modules of Figure 1 were used in our TREC QA submission. The remaining components, shown in white, were integrated after TREC. The next subsections describe the implementation of the modules comprising the system, focusing first on those which were part of the TREC evaluation, followed by a description of our post-TREC extensions.

2.1 Components Used in the TREC Evaluation

2.1.1 Question Analysis

The Question Analyzer (QA) is responsible for producing an initial analysis of the question text on which the rest of the system, including the Planner, bases its processing. Given a question sentence as its input, the QA generates a *request object*, a semi-structured representation of the question containing: the classification of the question according to a predefined taxonomy of question and answer types; a list of keywords, their types (either word, phrase, or proper-name), and alternate forms; answer-type specific constraints and features used during answer candidate extraction; and an f-structure representation of the question.

Central to the entire system is our question type and answer type taxonomy, based on [7] [13] [9]. In JAVELIN, the question type is used to guide the overall planning strategy and the answer type specifies a semantic category for the desired answer. Examples of this classification are presented in Table 1. At the time of TREC, the top-level answer type categories recognized by JAVELIN were: proper-name, temporal, location, numeric-expression, and a default object category. These represent a small fraction of the answer types the system will eventually include.

The Question Analyzer relies on both word-based linguistic information and sentence-level syntactic analysis to construct the request object. A word-level analysis of the question tokens is produced by combining information from several external resources, namely Wordnet [5] (for word morphology and semantic categorization), the Brill part-of-speech tagger [2], BBN IdentiFinder [1] , and the KANTOO Lexifier [14]. Once the word-level analysis is complete, pattern-matching is used to assign a question and answer type, and to identify answer-type specific constraints. The results of the word-level analysis and classification are then passed to the KANTOO GLR Syntaxifier [14] to create the f-structure.

2.1.2 Document Retrieval

The main function of the Retrieval Strategist (RS) module is to identify and retrieve documents that are likely to contain an answer to the current question. As a secondary function, the RS module also fulfills any other document repository requests made by individual system components, such as requests from the Information Extractor for specific documents or passages.

Internally, the Retrieval Strategist uses the Inquery retrieval system [3]. Stemming is done at indexing time using the Wordnet morphology library. Prior to indexing, source documents are preprocessed with the BBN IdentiFinder named-entity tagger [1], which tags the following entity types: Organization, Time, Date, Person, Place, Name, Currency, Amount, Number, Percentage. This analysis attempts to focus subsequent retrieval on documents containing not only the relevant keywords, but relevant data types. At indexing time, any terms within a span of text identified as a named-entity are stored in the index using a corresponding set of special fields. We also use a special extension to Inquery to recognize numeric expressions.

Table 1: Examples of JAVELIN's question and answer type classification.

Question	Question Type	Answer Type
Who invented the paper clip?	event-completion	proper-name
When was Hurricane Hugo?	time-expression	temporal
Where is the Danube?	location-expression	location
What is the size of Argentina?	feature-completion	numeric-expression
What did Vasco da Gama discover?	event-completion	object

Document retrieval requests sent to the RS consist of two main parts: the request object produced by the Question Analyzer, and a set of processing constraints (upper and lower limits on the number of documents to retrieve and a time limit). The Retrieval Strategist uses the keywords and answer type information supplied in the request object to construct an initial query. Keywords are included verbatim, and although the RS module does not currently perform any keyword expansion internally, it treats any alternate forms specified for the keywords as synonyms for retrieval. The likely answer type is mapped to a set of named-entity types corresponding to those used by the BBN IdentiFinder named-entity tagger. For example, a 'temporal' answer type is mapped to either a 'Time' or 'Date' named-entity tag. These types are treated as special keywords included in the terms passed to the retrieval system.

The document retrieval algorithm proceeds using an incremental query relaxation technique. The initial query is highly constrained, looking for all the keyword terms and data types in close proximity to each other. At each subsequent iteration, the algorithm relaxes one or more parameters in the query, such as the word proximity window. This assumes that documents containing answers will contain clusters of keywords and data types in closer proximity. The algorithm terminates if the number of documents retrieved is equal to the retrieval limit, or no additional relaxation steps are possible. The relaxation parameters are:

- The Inquery proximity/belief operator used to combine keywords. At each relaxation step, we either keep the same operator but expand the window size, or start with a new, more general operator. The operator applies to all keywords given in the query. For example, initially all keywords must be found within a proximity of three words. We then relax the operator to consider unordered 20-, 100-, and 250- word windows, followed by document-wide probabilistic AND, and so on.

- Phrase proximity, for any phrase keywords. This is usually kept at 3 words or less, until later in the relaxation regime, when it is expanded to the PHRASE operator.

- Proper name proximity, for any proper name keywords. This is usually kept at 3 words or less until very late in the relaxation regime, when it is expanded to the PHRASE operator.

- The inclusion or exclusion of the special named-entity keywords corresponding to the answer type. This alternates between on and off at every relaxation step.

The RS module output is a ranked list of document IDs. For some answer types, such as numerical types, offset information is provided with each document ID to simplify subsequent processing.

2.1.3 Answer Candidate Extraction

The Information Extractor (IX) is responsible for identifying and scoring answers from a set of potentially relevant documents. The goal of the IX is to find small relevant passages, identify the candidate answer in each such passage, and score each $\{passage, answer\}$ pair by estimating the probability that it answers the original question. The IX takes as input: a set of documents, the request object produced by the Question Analyzer, and processing constraints such as the minimum number of passages to be extracted or the time limit for the task. The IX output consists of passage-answer-score triplets, where the score is a measure of the degree to which the passages and answers solve the original question. The first step in information extraction is a loose passage filter which considers all passages that meet a minimum requirement based on a relaxed version of the task. This step produces a collection of possible $\{answer, passage\}$ pairs from the document set. Then the IX computes a set of features for each $\{passage, answer\}$ pair. These features are then used by a classifier to assign relevance scores to each answer candidate.

Currently, the features supplied to the classifier are based on the surface form and surface statistics of the passages, and make use of part-of-speech analysis, named-entity tagging, and morphological normalization. These features identify patterns and check for the existence of various cues that indicate whether the $\{passage, answer\}$ under scrutiny is relevant to the original question. For example, features may include patterns such as "QNOUN was QVERB in | on | at ANOUN", surface statistics such as the number of query terms in a given passage, or a measure of punctuation occurring between the query terms and the answer term. Given such features, the classifiers are trained off-line and tuned for better generalization.

There are currently two versions of the IX module, each based on a different classifier: K-Nearest-Neighbor (KNN) implementing KD-trees, and a decision tree using the $c4.5$ algorithm. For the KNN version, a parameter optimization was performed in Matlab yielding the number of nearest neighbors per query $nn = 25$, positive neighbor emphasis $\alpha = 1.7$, and exponent for the nearest neighbors $\beta = 1.5$.

2.1.4 Answer Selection

The Answer Generator (AG) module's purpose is to produce a list of answer candidates sorted by confidence score. Its input is a list of potential answer candidates, their associated scores assigned by the IX, and the passages they were extracted from. The basic algorithm used by the AG is as follows: the potential answer candidates are put into canonical form, and their scores are normalized to $[0, 1]$. Then all but the highest ranked candidate from each document is removed.

The canonical form for a given answer depends on the answer type. For TREC, location, numeric, time and person-name answer candidates were canonicalized into type-specific formats, while all other types were normalized simply by removing punctuation and converting to lowercase. A numeric answer is canonicalized to a pair containing the unit, which may be empty, and the value. A location, such as a country, is converted to the standard short form as specified in the CIA World Factbook and converted to lowercase. Dates are cast in *mm/dd/yyyy* format with unknown pieces marked. Names are split into first and last names, either possibly empty, and converted to lowercase.

The confidence score normalization is accomplished by taking the input confidence score range, $[0, 2200]$, using the frequency of answer confidence scores to fit a normal distribution over this range, and directly mapping all values in $[-2\sigma, 2\sigma]$ to $[0, 1]$. All outlying confidence values are set to 0 and 1 respectively.

The answers are then grouped into clusters, where each member of the cluster supports the most specific member of the same cluster. As with canonicalization, the definition of supporting depends on the answer type. For example, a numeric answer A would support numeric answer B if A and B have the same units and $\frac{|A-B|}{B} < .05$, while a person-name answer would support another if they had the same last name.

The confidence scores for an answer cluster are then computed as the probability that at least one member of the cluster is correct given that all answers in the cluster are independent and equally weighted. For a cluster C containing answers $A_1, A_2, ..., A_n$ with scores $S_1, S_2, ..., S_n$, the confidence for the entire cluster T_C are computed with the following formula:

$$T_C = 1 - \Pi_{i=1}^{n}(1 - S_i) \qquad (1)$$

2.1.5 Execution Coordination

For the TREC evaluation, the Execution Manager was used to coordinate batch processing of the test set, calling each of the four components in a fixed sequence to imitate a pipelined architecture.[1] At each step, the EM constructs the module-specific input XML, calls the next module in the sequence with that input, and stores the resulting module output in the system Repository for later use.

2.1.6 Data Storage

The Repository is the information backbone of the system and stores all the information produced by the QA modules. The repository module consists of a relational database and a file system database. The relational database was developed using Microsoft SQL 2000, and currently holds 45 tables and over 3 million records.

2.2 Post-TREC Extensions

2.2.1 Planning

One of the major extensions to the JAVELIN system subsequent to the TREC submission was to integrate the Planner module, which now serves as the controller for the question answering process. The Planner is responsible for selecting and issuing module command sequences to maximize the expected utility of the information JAVELIN produces, taking into account the available system resources and constraints such as

[1] In the complete system, the EM relies on the Planner module to determine module sequencing, and simply acts as a broker between the Planner and the other modules in the system. It translates individual Planner requests into the input XML required by the requested module, saves results and process history information in the Repository, and provides user authentication.

total execution time. The primary advantage planning offers is the flexibility to dynamically select between different versions of the system components, enabling JAVELIN to generate different QA strategies at runtime, rather than relying on a fixed pipeline architecture.

The Planner operates as a service for the JAVELIN GUI, and communicates with the rest of the system via the Execution Manager. Upon receiving a new question, the Planner calls the Question Analyzer via the EM to perform the initial question analysis, from which it generates a planning problem describing the initial state and information goal. The Planner then begins the planning and execution process, continuing until it has met the goal criteria (has found an answer or set of answers with sufficiently high expected utility), or has exhausted its available resources. At this point, it returns the answer or a failure message. The Planner module also provides an "interactive" mode, which allows user feedback during planning.

Internally, the Planner uses a forward-chaining utility-based planning and execution algorithm that performs a best-first search across the set of possible information states [10]. It is supplied with a domain model describing the features of the information state on which the planning process will be based, and the actions the Planner can select between, namely the various modules that comprise the system, the preconditions under which they are applicable, and their possible effect on the information state. It is also given a problem statement consisting of: an initial state, a predefined utility-function, a utility success threshold, and a value specifying a confidence threshold for termination. Beginning with the initial state and an empty plan, the Planner evaluates the successor states of each candidate action, selecting the one with the highest expected utility to add to the partial plan. The internal planning state is updated to reflect the projected outcome, and the action selection process repeats. At each step, the algorithm considers the tradeoff between executing the first unexecuted action in the plan, and continuing to plan with the uncertainty of the projected states. If an execution step is carried out, it is followed by an assessment of the need for replanning. The algorithm terminates when all steps in the plan have been executed and the confidence and utility thresholds for goal satisfaction are met, or there are no additional actions the Planner can take.

2.2.2 Answer Justification

The Answer Justification module produces an audit trail of the processing performed by JAVELIN during the course of answering a question. The purpose of this audit trail is twofold: first, it supplies evidence regarding an answer's correctness, and second, it documents the processing decisions made by the system. Our eventual goal is to use this module to enable a user to interactively provide feedback to the system, help guide processing choices, or correct system knowledge errors as they arise.

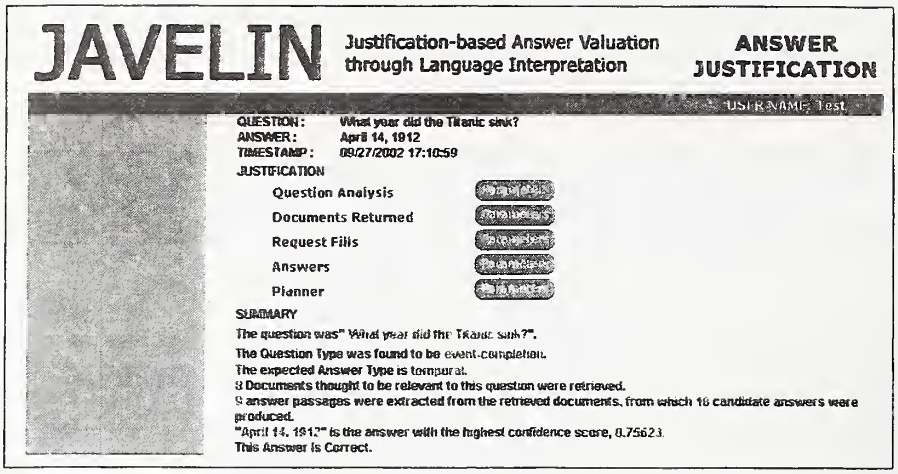
The current beta version of the Answer Justification module provides the user with a read-only display containing a brief summary of the information produced during processing, and a detailed trace of each step of the execution. Question summaries are generated automatically from the repository data, and include: the question and answer type classification assigned by the system, the number of documents and answer candidates generated, the highest ranked answer and its associated confidence score, and when applicable, the associated TREC relevance judgement. A sample summary is shown in Figure 2.

2.2.3 GUI

The JAVELIN GUI (Figure 3) is the front-end to the Planner-driven system. It is a Java application that resides on the user's desktop, forwarding user requests to the Planner module, and displaying the resulting answer(s). The GUI also provides a mechanism for the Planner to interactively request feedback from the user, by displaying Planner-initiated dialogs and returning user responses.

3 TREC Results

A preliminary version of JAVELIN without the Planner module and user-interface components was tested on the TREC QA track evaluation data in July 2002. This version used the Execution Manager to invoke each of the four major components in the following fixed sequence: the Question Analyzer, the Retrieval Strategist, the Information Extractor, and the Answer Generator. The goal of this initial test was to provide us with a baseline for subsequent evaluation of the complete system with the Planner and improved components.



JAVELIN
Justification-based Answer Valuation through Language Interpretation
ANSWER JUSTIFICATION
USER NAME: Test

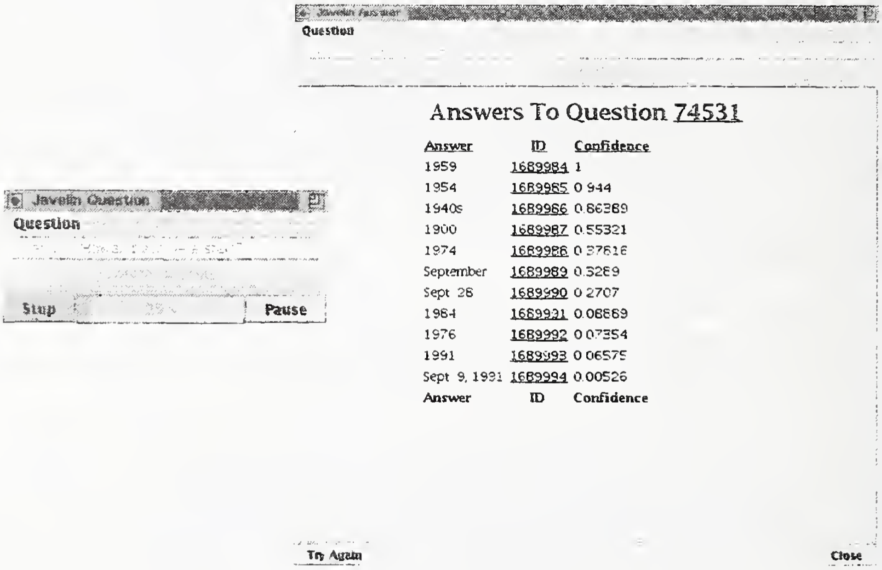
QUESTION: What year did the Titanic sink?
ANSWER: April 14, 1912
TIMESTAMP: 09/27/2002 17:10:59

JUSTIFICATION

SUMMARY

The question was" What year did the Titanic sink?".
The Question Type was found to be event-completion.
The expected Answer Type is temporal.
8 Documents thought to be relevant to this question were retrieved.
9 answer passages were extracted from the retrieved documents, from which 18 candidate answers were produced.
"April 14, 1912" is the answer with the highest confidence score, 0.75623.
This Answer Is Correct.

Figure 2: Screenshot of the output produced by the beta version of the Answer Justification module.



Answers To Question 74531

Answer	ID	Confidence
1959	1689984	1
1954	1689985	0.944
1940s	1689986	0.86389
1900	1689987	0.55321
1974	1689986	0.37616
September	1689989	0.3289
Sept 28	1689990	0.2707
1964	1689991	0.08869
1976	1689992	0.07354
1991	1689993	0.06575
Sept 9, 1991	1689994	0.00526
Answer	ID	Confidence

Figure 3: JAVELIN GUI: main control window and resulting answers.

Two TREC QA runs were submitted, one using the decision tree version of the Information Extractor, and the other using the KNN version to identify candidate answer passages. In each run, the system considered only the top 15 documents returned by the Retrieval Strategist module. Table 2 summarizes our official TREC results for these runs. The KNN run produced more correct answers than the DT run (86 vs. 75), but the DT run received the higher weighted score once the system's confidence estimates were taken into account (0.251 vs. 0.209). Both runs exhibited comparable precision in identifying questions without an answer (0.152 for the DT, 0.164 for the KNN run).

It is premature to draw conclusions about the relative performance of the two classifiers, given the limited size of the training and test sets. However, this preliminary evaluation did enable us to make several observations about the system's performance in general. The next section describes our post-TREC analysis.

Table 2: JAVELIN TREC11 QA results using DT-based (CMUJAV000495) and KNN-based (CMUJAV000501) answer candidate identification, with retrieval of the top 15 documents.

Submitted Run	CMUJAV000495 (DT)	CMUJAV000501 (KNN)
Correct (R)	75	86
Inexact (X)	13	12
Unsupported (U)	10	8
Wrong (W)	402	394
Confidence-weighted score	0.251	0.209
No-answer precision	(12/79) 0.152	(10/61) 0.164
No-answer recall	(12/46) 0.261	(10/46) 0.217

Table 3: Analysis of individual module performance for a subset of 193 questions from TREC 2002.

	Module	Total Questions	Exceptions	Correct	Possibly Correct	Incorrect
	EM	193	0	192 (99.5%; 99.5%)	NA	1 (0.5%; 0.5%)
	QA	192	0	154 (80.2%; 79.8%)	23 (12.0%; 11.9%)	15 (7.8%; 7.8%)
	RS	177	0	80 (45.2%; 41.5%)	70 (39.6%; 36.3%)	27 (15.3%; 14.0%)
KNN	IX unique tied	150	8 (5.3%; 4.2%)	37 (24.7%; 19.2%) 39 (26.0%; 20.2%)	82 (54.7%; 42.5%) 80 (53.3%; 41.5%)	23 (15.3%; 11.9%)
	AG	119	0	47 (39.5%; 24.4%)	66 (55.5%; 34.2%)	6 (5.0%; 3.1%)
DT	IX unique tied	150	14 (9.3%; 7.3%)	36 (24.0%; 18.7%) 60 (40.0%; 31.1%)	74 (49.3%; 38.3%) 50 (33.3%; 25.9%)	26 (17.3%; 13.5%)
	AG	110	0	41 (37.3%; 21.2%)	66 (60.0%; 34.2%)	3 (2.7%; 1.6%)

4 Analysis

In the interim between performing the TREC QA evaluation and receiving our official scores, we conducted an internal performance analysis for a subset of the TREC 2002 question set. Project members manually identified correct answers for 193 questions, along with at least one document containing the answer. We then compared our manually generated answer key with the system's output to determine whether or not the system returned the correct answer, and if not, at what step the first failure occurred.

Table 3 summarizes the results of this analysis. For each module, we computed the number of questions where a module exception occurred, the number for which the module performed correctly, the number for which it might have performed correctly, and the number that resulted in erroneous output. The total number of input questions (column 2) for each subsequent module excludes any for which an error or exception occurred earlier in the pipeline. The "correct" and "possibly correct" columns distinguish between completely correct analysis and partially correct analysis produced by the Question Analyzer, and cases where the output of the RS/IX/AG module included the correct information, but did not assign it the highest rank. Additionally, because it was possible for multiple answers to be assigned the same confidence by the IX module, we distinguished between cases where the correct answer was uniquely ranked first, and cases where the correct answer was one of several answers receiving the same high score. The fraction of the questions covered by each outcome is provided in parentheses as a relative percentage of the input questions for that module, and as an absolute percentage of the 193 questions we evaluated.

Errors occurring during question analysis were due primarily to insufficient coverage of classification patterns for question and answer type classification. This was a known issue going into the evaluation, given the limited number of answer types implemented, and is being addressed in our ongoing development.

Roughly equal percentages of the failures occurred during document retrieval and candidate extraction. For approximately 15% of the good input that the RS received, it failed to retrieve any documents (within the top 15) containing the correct answer. Likewise, in 15 – 17% of the cases where at least one document containing the answer was passed to the Information Extractor, the IX failed to identify it as a candidate. Certainly, a significant number of these errors were due to the limited number of answer types recognized by the system. Our default strategy of assigning an "object" answer type provides very little additional information (beyond that provided by the keywords) for the Retrieval Strategist or Information Extractor to use in discriminating between documents and candidates respectively. The impact on the RS is simply a restriction on its ability to augment queries with named-entity tags. However, in the IX, where extraction and candidate scoring relies on answer-type dependent features, the impact is large.

In cases where the correct answer was included amongst the candidates passed to the Answer Generator, the clustering algorithm of the AG was sometimes able to compensate for IX ranking inaccuracies. In the KNN run, the Answer Generator successfully identified the correct answer in 9.5% of the cases where the Information Extractor did not rank the correct answer highest or gave it a tied ranking for the highest score. In the DT run, the Answer Generator was able to correctly compensate for IX ranking errors approximately 5% of the time. This smaller compensation is likely due to the fact that the decision-tree classifier produces less discriminatory (coarser-grained) confidence estimates. The Answer Generator failed to rank the correct answer highest for answer candidate sets containing hundreds of unique candidates, suggesting a need for better filtering mechanisms and more sophisticated methods of combining evidence.

To evaluate how much better our retrieval performance could potentially be using our existing query strategy and just increasing the number of documents retrieved, we computed the probability that the RS returns at least one correct document within the top N documents for several values of N. Table 4 presents the retrieval success rates for several values of N ranging from 15-120, using the previously analyzed subset of the TREC 2002 questions as the test set. Going from 15 documents to 60 documents increased the likelihood of retrieving a correct document by 11%, but doubling the number of documents again to 120 added very little. Given the negative impact that a larger retrieval set has on processing time, and the potential for increased noise in the answer candidate set, our use of 15 documents appears to be a reasonable trade-off between coverage and performance. We are now exploring constrained query expansion to improve our retrieval success rate.

The failure rate of the Information Extractor for the TREC 2002 subset was consistent with its performance on previous TREC question sets. Table 5 summarizes the performance of the DT version of the IX module on past TREC data. The first column contains the number of questions in the test set, the second shows the number of times the correct answer was among the top five highest score answer produced by the IX, and the last column represents the number of times the correct answer existed in at least one document provided by the Retrieval Strategist. Setting aside the issue of answer type coverage, these results and an examination of individual failures suggest that more versatile extraction strategies are required in order to locate the right candidate answer more often. We are currently in the process of augmenting the system with an NLP-based version of the IX to supplement our current statistical approaches.

Table 4: Retrieval Strategist success rates for various values of top N documents retrieved

N	Success Rate
15	0.74
30	0.80
60	0.85
120	0.86

Table 5: Performance of the Information Extraction module on TREC question sets

Data	Corpus Size	$A\ inA^5$	$A \subset D$
TREC 8	200	71	189
TREC 9	693	218	424
TREC 10	500	119	313

5 Discussion

The JAVELIN system submitted to TREC is an integrated architecture for open-domain question answering. JAVELIN's modular approach addresses individual facets of the QA task with different modules. Question analysis addresses the taxonomy of question-answering types and type specific constraints by combining knowledge and pattern matching. Document retrieval includes query processing, document retrieval, and passage retrieval, and implements a strategy of incremental query relaxation. The information extraction module employs decision trees and KNN classifiers to identify answer passages in the relevant documents. Answer selection includes type normalization, conversion, clustering, and prototype selection. The behavior of the individual modules is coordinated by the Planner module, which controls the overall question answering process. The Execution Manager handles inter-module communication, persistent data storage and retrieval, and authentication while servicing requests made by the Planner. The Repository component supports the entire system with a large-scale, centralized data and state storage capability.

Using the decision tree classifier, the system obtained a confidence weighted score of 0.251; with the KNN classifier the system obtained a score of 0.209. These results suggest several improvements for each module. Question analysis needs to perform deeper NLP processing. Document retrieval requires a more complex query expansion to improve the module's success rate. The TREC results also reflect the fact that the feature space used by the classifiers is limited; a richer set of features and more training data are likely to improve performance. We also plan to improve answer selection by using external knowledge and additional local constraints to combine multiple answers from multiple sources.

References

[1] BBN Technologies. *IdentiFinder User Manual*, 2000.

[2] E. Brill. Unsupervised learning of disambiguation rules for part of speech tagging. In David Yarovsky and Kenneth Church, editors, *Proceedings of the Third Workshop on Very Large Corpora*, pages 1–13, Somerset, New Jersey, 1995. Association for Computational Linguistics.

[3] J. P. Callan, W. B. Croft, and S. M. Harding. The INQUERY retrieval system. In *Proceedings of the Third International Conference on Database and Expert Systems Applications*, pages 78–83, Valencia, Spain, 1992.

[4] C. Clarke, G. Cormack, D. Kisman, and R. Lynam. Question answering by passage selection. In *Proceedings of the 9th Text Retrieval Conference (TREC-9)*, 2000.

[5] C. Fellbaum. *WordNet - An Electronic Lexical Database.* Cambridge, Mass : MIT Press, 1998.

[6] O. Ferret, B. Grau, M. Hurault-Plantet, G. Illouz, C. Jacquemin, N. Masson, and P. Lecuyer. QALC - the question answering system of LIMSI-CNRS. In *Proceedings of the 9th Text Retrieval Conference (TREC-9)*, 2000.

[7] A. C. Graesser, N. Person, and J. Huber. *Mechanisms that Generate Questions*, chapter 9, pages 167–187. Lawrence Erlbaum Associates, 1992.

[8] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu. FALCON: Boosting knowledge for answer engines. In *Proceedings of The Ninth Text REtrieval Conference (TREC 9)*, 2000.

[9] L. Hiyakumoto. Planning and execution for open-domain question answering. Thesis proposal, November 2001.

[10] L. Hiyakumoto and M. Veloso. Towards planning and execution for information retrieval. In *Proceedings of the AIPS'02 Workshop on Exploring Real-World Planning*, Toulouse, France, 2002.

[11] E. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C-Y. Lin. Question answering in webclopedia. In *Proceedings of the 9th Text Retrieval Conference (TREC-9)*, 2000.

[12] K. Kwok, L. Grunfeld, N. Dinstl, and M. Chan. TREC-9 cross language, web and question-answering track experiments using PIRCS. In *Proceedings of the 9th Text Retrieval Conference (TREC-9)*, 2000.

[13] W. G. Lehnert. *The Process of Question Answering.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1978.

[14] T. Mitamura and E. Nyberg. The KANTOO machine translation environment. In *Proceedings of AMTA-2000*, 2000.

[15] J. Prager, E. Brown, D.R. Radev, and K. Czuba. One search engine or two for question-answering. In *Proceedings of the 9th Text Retrieval Conference (TREC-9)*, 2000.

ICT Experiments in TREC-11 QA Main Task

Hongbo Xu, Hao Zhang, Shuo Bai
Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
hbxu@software.ict.ac.cn
http://www.ict.ac.cn/

Abstract

This is the second time we participate in the TREC-QA track. We put emphasis on candidate passage ranking and answer matching. As to named entity tagging, we applied the latest version of GATE and did some succeeding work aiming at our goal. This paper presents our methods in detail.

Keywords: TREC-QA, candidate passage ranking, answer matching

1. Introduction

We took part in the TREC-QA track for the second time this year. Of the main and list subtasks, we still undertook the main subtask. Three QA runs have been submitted for evaluation.

The document set for TREC-11 QA track has been changed to the new AQUAINT disk set released by AQUAINT Data Set Organization. The QA main task has several differences from previous years' tasks. Each question requires exactly one response, and the question set should be ordered by confidence in the response. The score assigned to each question will be 1 if the judgment is correct, and 0 otherwise. A measure that is an analogue to document retrieval's uninterpolated average precision will be used to score the run as a whole. The measure is computed as:

[sum for i=1 to 500 (#-correct-up-to-question-i/i)] / 500

Obviously, this measure will reward systems that correctly rank questions it answered correctly before questions it answered incorrectly.

Inspired by experiments on web track, we changed the weighting method of SMART to meet the need of TREC documents. We've made many experiments on candidate passages ranking to seek a better method and proper parameters. Another focus of our efforts is to improve the precision of answer extracting and matching. Aiming at the measure of TREC-11, we give priority to questions with types that were processed well in last year.

2. System Description

Our TREC-11 QA system is based on last year's. SMART[2], pairing sentences module, candidate passage ranking module and GATE[2] are used to retrieve relevant documents from data set and produce ranked named entities as candidate answers. Question analyzer analyses every question to identify the question type and keywords. Answer extracting and matching module matches the question type with the named entities. Answer outputting module outputs the most credible named entity and orders the question set by confidence in the answer. Figure 2.1 illustrates the whole architecture.

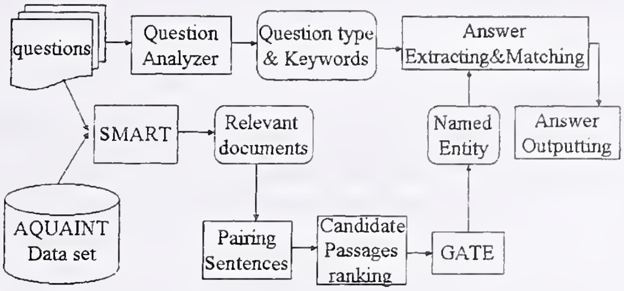


Figure 2.1: Architecture of the TREC-11 main QA System

3. Pairing sentences and Candidate passage ranking

In experiments on web track, we find that the weighting method of SMART is not so fit for TREC documents, so we modify its weighting module, taking *(log(tf)+1)*idf* instead of *tf*idf.* Then we use the revised SMART to retrieve 50 relevant documents for each question from the AQUAINT data set.

Subsequently, we parse these documents into sentences, and then assemble a candidate passage every two successive sentences that both have keywords in common with the question. The algorithm presented last year in section 4.4 of paper[2] is still used to rank the candidate passages for each question. Our new experiments show that step6 of the algorithm is of little help. So we devise a new method as an alternative. The score added to a candidate passage P is computed by:

$$\beta * count\_m / (count\_q + count\_k)$$

Where *count_m* is the number of matching keywords between the question and the candidate passage P, *count_q* the number of keywords in the question and *count_k* the number of keywords in P. β is an experiential parameter.

To lighten the burden of GATE in next process, for each question we only reserve the top 10 or 20 ranked candidate passages for named entity tagging.

4. Answer extracting, matching and outputting

We still use GATE as our Named Entity tagger. The latest version of GATE 2.0 (*released on March 15, 2002*) realizes the function to process a document set serially, which not only saves the time on loading modules when processing, but also allows us processing more candidate passages for one question. GATE 2.0 also optimizes the identification of type LOCATION, PERCENT, ORGANIZATION and PERSON. As to type NUMBER and MEASUREMENT, we still need to take some succeeding steps to assemble an integrated NUMBER or MEASUREMENT entity.

As in last year, we use a question analyzer to identify the question type and keywords of each question by two kinds of rules: keyword-based and template-based[2]. The main difference from last year is that we've made a consistency check on these rules to eliminate the collision when applying them.

Answer extracting, matching and outputting module compares the question type with the named entities in candidate passages and chooses the most credible named entity as final output.

To optimize for TREC-11 measure, we should order the question set by confidence in the answer. Without experiments supporting, we intuitively give priority to questions with types that were processed well by our system in last year.

5. Results and Analysis

We have submitted three runs for QA main task. In *ICTQA11a* and *ICTQA11b* we produce question answers from the top 10 candidate passages, the difference between them is that the candidate passages are ranked with different strategies. *ICTQA11b* and *ICTQA11c* have the same ranking strategies. But in *ICTQA11c*, question answers are derived from the top 20 candidate passages. Table 5.1 shows the evaluation results.

RunID	Confidence-weighted score	Wrong #	Unsupported #	Inexact #	Right #	Precision of recognizing no answer	Recall of recognizing no answer
ICTQA11a	0.091	445	9	4	42	10/58 =0.172	10/46 =0.217
ICTQA11b	0.084	440	7	6	47	9/69 =0.130	9/46 =0.196
ICTQA11c	0.088	435	8	9	48	9/47 =0.191	9/46 =0.196

Table 5.1 Statistics over all 500 questions of our runs in TREC-11

Our system does badly on most question types, except that the No Answer, DATE and LOCATION types are done a little better. Since only one response is allowed for each question, we think it's the simple answer matching strategy that does so much harm to the performance. Lack of time, many experiments aborted, so we had to give up trying our new answer matching methods. In addition, applying syntactic and semantic parsing technique should be a good approach to solve the problem on answer extracting and matching.

6. Conclusions

We've participated in the TREC-QA track for two times. By communicating with friends from China and abroad, we've learned much. We've also realized that there is a long way for us to go on QA research. But we are sure to do better in the future.

Acknowledgements

This research is supported by the national 973 fundamental research program under contact of G1998030413, the Institute Youth Fund under contact 20016280-9 and the Institute Youth Fund under contact 20026180-24. We give our thanks to all the people who have contributed to this research and development, in particular Bin Wang, Xueqi Cheng, Yitao Zhang, Sujian Li and Huaping Zhang.

[1] E. Voorhees, Overview of the TREC 2001 Question Answering Track. In *The Tenth Text REtrieval Conference (TREC 10)*, page 42, 2001.

[2] B. Wang, H. Xu, Z. Yang, Y. Liu, X. Cheng, D. Bu, S. Bai. TREC-10 Experiments at CAS-ICT: Filtering, Web and QA. In *The Tenth Text REtrieval Conference (TREC 10)*, page 109, 2001.

[3] M.M. Soubbotin, Patterns of Potential Answer Expressions as Clues to the Right Answers, In *The Tenth Text REtrieval Conference (TREC 10)*, page 293, 2001.

[4] S. Alpha, P. Dixon, C. Liao, C. Yang, Oracle at TREC 10: Filtering and Question-Answering, In *The Tenth Text REtrieval Conference (TREC 10)*, page 423, 2001.

TREC-11 Experiments at CAS-ICT: Filtering and Web

Hongbo Xu. Zhifeng Yang. Bin Wang. Bin Liu. Jun Cheng, Yue Liu, Zhe Yang, Xueqi Cheng. Shuo Bai

Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China

{wangbin,hbxu,zfyang,yliu,cxq,yangzhe,bai}@ict.ac.cn

http://www.ict.ac.cn/

Abstract

CAS-ICT took part in the TREC conference for the second time this year and we undertook two tracks of TREC-11. For filtering track, we have submitted results of all three subtasks. In adaptive filtering, we paid more attention to undetermined documents processing, profile building and adaptation. In batch filtering and routing, a centroid-based classifier is used with preprocessed samples. For Web track, we have submitted results of both two subtasks. Different factors are considered to improve the overall performance of our Web systems. This paper describes our methods in detail.

Keywords: TREC-11, Filtering, Web track

1. Introduction

CAS-ICT took part in the TREC conference for the second time this year, and we have submitted results of filtering track and Web track.

For filtering track, we undertook all three subtasks. Our adaptive filtering system is still based on VSM. Our Rocchio-like profile adaptation algorithm puts stress on the undetermined documents and some strategies are proposed for T11U or T11F optimization. Four runs have been submitted for evaluation: all of them are optimized for T11U measure, but in three of them T11F measure is also considered at the same time. In batch filtering and routing, we use a centroid-based classifier with preprocessed samples. Two batch filtering runs and two routing runs have been submitted for evaluation. In all of our filtering experiments, we do not use any other resources except the New Reuters Corpus.

For Web track, we undertook both the Named Page Finding task and the Topic Distillation task. Our system is based on SMART(*ftp://ftp.cs.cornell.edu/pub/smart*). In the former task, we try to integrate different factors to improve the overall system performance. In the latter task. a variant HITS algorithm is used to the top *n* results returned by SMART. Five Named Page Finding results and three Topic Distillation results have been submitted for evaluation.

2. Filtering

For filtering track, we undertook all three subtasks, but we paid more attention to the adaptive filtering task. Batch filtering and routing tasks are used to test our new classifier.

2.1 Adaptive Filtering

2.1.1 Introduction

The total 100 topics used in the filtering task this year can be divided into two sets: the first 50(R101-R150) topics are called assessor topics, which are hand-built by NIST assessors. and the last 50(R151-R200) topics are called intersection topics, which are derived from Reuters category intersections. The two sets have been evaluated separately.

New Reuters Corpus (http://about.reuters.com/researchandstandards/corpus/) is still used this year, but the training set and testing set are different with TREC-10. The first 83,650 documents are used for training (training set) and the remaining about 720,000 documents for testing (testing set). The official adaptive filtering measures are utility (T11U, scaled using Ault's formula), and F-beta (T11F, beta = 0.5). The former is a linear utility measure and the latter is a kind of F-measure. Additionally, set precision and set recall measures are also reported in the final results. In the adaptive subtask, only three positive samples in training set are given for each topic, and the goal is to retrieve relevant documents one by one from the coming testing documents stream and get maximum T11U or T11F value at the same time.

2.1.2 System Description

Last year, we have built an adaptive filtering system, which consists of two components: the profile initialization component and the profile adaptation. This year we made some improvement based on this system, in particular, in the profile initialization and optimization modules.

2.1.3 Initialization

Our initialization process includes common operations such as term tokenization. stop words elimination, stemming, *TF* and *IDF* computation. Each topic is treated as a document and processed in the same way. The initial profile vector can be obtained by summing up the topic vector and the three positive documents vectors with different weight. Meanwhile, we set the initial threshold by computing the similarities between the initial profile and all the documents in the training set.

Since we can’t use the *IDF* statistics of testing set till now, we take the *IDF* statistics of the training set as an alternative for term weighting. Ideally, we should update the *IDF* statistics when retrieving new documents from the testing documents stream. But our previous experiments have indicated that it does not seem to improve the overall filtering performance. Therefore, we use the *IDF* statistics of the training set without any modification all over our experiments.

Term selection

Last year, we applied a new method for feature selection, which can be regarded as a variation of Mutual Information. The final results indicated that our method is successful when the topic is a single Reuters category.

However. each topic of this year has been changed into a natural language statement or an intersection of some Reuters categories. Our experiment shows that the method does not work well this year. Several experiments show that the simple term selection according tu the *TF* and *DF* values is a good choice.

Profile initialization

For each topic. the profile vector (denoted as $\vec{P}$) is the weighted sum of the topic vector (denoted as $\vec{T}$) and the feature vector (denoted as $\vec{F}$), which is the sum of the initial three positive documents vectors. The formula is:

$$\vec{P} = \alpha * \vec{F} + \beta * \vec{T} \qquad (2.1)$$

In our experiment. we set $\alpha=1,\beta=3$ to give prominence to the topic vector.

Similarity computation

We still use the vector *cosine* distance to compute the similarity between a profile vector ($\vec{P}_i$)

and a document vector($\bar{D}_j$). *TFIDF* value is used in our system, which is computed by $(\log(TF_i)+1)*\log(1+\frac{N}{DF_i})$, where *N* is the number of the total documents in the training set.

2.1.4 Adaptation

For each topic, after initializing the profile and the threshold, we can scan documents one by one from the testing set. If the similarity between the profile and the document is higher than the threshold, the document is retrieved, else not. Then we check the answer list of the testing set to find whether the document is really relevant or not. With this information, we can take some kind of adaptation to improve the system performance. The adaptation may include threshold updating and profile updating.

Threshold adaptation

As we know, the goal of the TREC-11 adaptive filtering system is to get maximum T11U or T11F. Therefore, we adjust the threshold for T11U optimization or T11F optimization.

For T11U, our direct goal is to avoid negative utility value for each topic. When the utility value becomes negative during filtering, which means the system retrieves too many non-relevant documents, we augment the threshold to reduce the number of retrieved documents. Another optimization strategy we take is to improve the precision while the recall can't be greatly reduced.

For T11F, our goal is to avoid retrieving zero "relevant" documents. We reduce the threshold when the system retrieves zero documents at an interval.

Profile adaptation

As the filtering task indicates, each profile vector represents a user's interest. After retrieving more and more relevant or non-relevant documents, we can get more and more useful information about the user's interest, which can help us adapt the profile. Our profile adaptation includes positive adaptation, negative adaptation and adaptation based on undetermined documents. For positive adaptation, we add the positive documents vectors to the old profile vector with weight α. For negative adaptation, we subtract the negative documents vectors from the old profile vector with weight *β*. For adaptation based on undetermined documents, we set a relative high threshold (we use *t*=0.6) to filter the retrieved undetermined documents. Those retrieved documents that have similarity below *t* are regarded as pseudo-negative documents and treated as real negative documents. A pseudo-negative document is used in negative with smaller β value. When retrieving the *n+1* th document D_{n+1}, we can adapt the *n*th profile to the *n+1* th profile according to the following formula:

$$\bar{P}_{n+1}=\begin{cases}\bar{P}_n+\alpha*\bar{D}_{n+1} & \text{if } D_{n+1} \text{ is relevant}\\ \bar{P}_n-\beta*\bar{D}_{n+1} & \text{if } D_{n+1} \text{ is irrelevant}\\ \bar{P}_n-\beta'*\bar{D}_{n+1} & \text{otherwise and } \mathrm{sim}(\bar{P}_n,\bar{D}_{n+1})<t\end{cases} \tag{2.2}$$

Thus after we have retrieved *n+1* documents, all the retrieved documents are divided into four sets: the relevant set denoted as $\{D^+\}$, the irrelevant set $\{D^-\}$, the undetermined but pseudo-negative set $\{D_u^-\}$ and the remaining documents set $\{D_u^+\}$. We do not use $\{D_u^+\}$ in the adaptation. Then the new profile vector is computed by:

$$\bar{P}_{n+1}=\bar{P}_0+\alpha*\sum_{D_i\in\{D^+\}}\bar{D}_i-\beta*\sum_{D_j\in\{D^-\}}\bar{D}_j-\beta'*\sum_{D_k\in\{D_u^-\}}\bar{D}_k \tag{2.3}$$

Formula (2.3) is some kind of the Rocchio[15] algorithm except one point: we do not compute the centroid of a document set and regard all documents in each set as one vector. In other words, we emphasize the retrieved documents and endow them the ability to adjust the profile vector quickly. As in last year, we investigate the values of α, β and β'. In our experiments, we set α=1, β=1.8 and β'=1.3.

Undetermined documents processing

In TREC-11, the relevance of most documents in the testing set is unknown to the system. In order to get more feedback information, we make some experiments on the undetermined documents.

Experiment 1: Ignoring the undetermined documents when filtering, we adjust the threshold only according to the relative proportion between the known relevant documents and irrelevant ones. But there is an important presupposition that such a distribution is the same in the undetermined documents. Unfortunately, we can' t prove this presupposition.

Experiment 2: A simple idea is that if we could know the real relevance of all documents in the testing set, the adaptation strategy proved effective in TREC-10 can still be applied. Therefore, we make a positive centroid and a negative centroid with the retrieved relevant and irrelevant documents during retrieving the testing set. When retrieving an undetermined document, we judge its relevance by computing its distance from the positive centroid and the negative centroid. Those undetermined documents that are nearer to the positive centroid will be treated as real relevant documents, while others will be treated as irrelevant documents. Thus we can simulate a situation as in TREC-10. This method allows the system retrieving plenty of "relevant" documents, which is helpful to the recall but against the precision. It seems that the initial values of the positive centroid and negative centroid greatly affect the judgment of undetermined documents. The positive centroid can be made by the known three positive samples, but we can't make a good negative centroid because we haven't any negative samples.

Experiment 3: Suppose the answer list has provided most real relevant documents in the testing set, we treat all or most of the undetermined documents as irrelevant documents. As we've introduced above, a threshold t can be used to filter the undetermined documents, those have similarity below t will be treated as irrelevant documents. The discussion of TREC-11 filtering mailing list shows that such a supposition is reasonable. With this method, we can control the retrieved "relevant" documents effectively, which is helpful to the precision. But when the number of real relevant documents in the testing set is big, such a system will suffer a heavy loss.

Of the three methods above, we apply the third one finally, partly suggested by the discussion of TREC-11 filtering mailing list. The results are encouraging.

2.1.5 Evaluation Results and Analysis

We have submitted four adaptive filtering runs: all for T11U optimization, in three of them we make balance between T11U and T11F. ICTAdaFT11Ud is optimized for recall, avoiding the heavy loss of relevant documents. As to the optimization method, we use local maximum optimization strategy at every adaptation interval to obtain the holistic maximum. We also adopt a method to avoid zero return at next interval by learning from the current adaptation interval.

Table 2.1 shows the results of the 50 assessor topics. Table 2.2 shows the results of the 50 intersection topics. Table 2.3 is the evaluation results of all 100 topics. Of the assessor topics, the system exhibits a good performance. But of the intersection topics, the system behaves badly.

Run ID	MeanT11U	T11U vs. median(topic nums)			MeanT11F	T11F vs. median(topic nums)		
		>(Best)	=	<(Worst/Zero)		>(Best)	=	<(Worst/Zero)
ICTAdaFT11Ua	0.475	46(6)	3	1(0/0)	0.427	43(5)	0	7(2/2)
ICTAdaFT11Ub	0.475	46(6)	3	1(0/0)	0.428	43(5)	0	7(2/2)
ICTAdaFT11Uc	0.471	45(6)	3	2(0/0)	0.422	41(4)	0	9(2/2)
ICTAdaFT11Fd	0.321	18(0)	2	30(3/3)	0.306	29(0)	2	19(2/2)

Table 2.1 ICT adaptive filtering runs(Assessor topics) in TREC-11

Run ID	MeanT11U	T11U vs. median(topic nums)			MeanT11F	T11F vs. median(topic nums)		
		>(Best)	=	<(Worst/Zero)		>(Best)	=	<(Worst/Zero)
ICTAdaFT11Ua	0.335	50(18)	0	0(0/0)	0.061	12(5)	32	6(6/6)
ICTAdaFT11Ub	0.330	49(17)	0	1(1/1)	0.062	13(3)	31	6(6/6)
ICTAdaFT11Uc	0.335	50(18)	0	0(0/0)	0.061	12(5)	32	6(6/6)
ICTAdaFT11Fd	0.240	19(0)	7	24(3/3)	0.052	21(1)	24	5(5/5)

Table 2.2 ICT adaptive filtering runs(Intersection topics) in TREC-11

Run ID	MeanT11U	T11U vs. median(topic nums)			MeanT11F	T11F vs. median(topic nums)		
		>(Best)	=	<(Worst/Zero)		>(Best)	=	<(Worst/Zero)
ICTAdaFT11Ua	0.405	96(24)	3	1(0/0)	0.244	55(10)	32	13(8/8)
ICTAdaFT11Ub	0.4025	95(23)	3	2(1/1)	0.245	56(8)	31	13(8/8)
ICTAdaFT11Uc	0.403	95(24)	3	2(0/0)	0.2415	53(9)	32	15(8/8)
ICTAdaFT11Fd	0.2805	37(0)	9	54(6/6)	0.179	50(1)	26	24(7/7)

Table 2.3 ICT adaptive filtering runs(all 100 topics) in TREC-11

We had partly noticed the problem of intersection topic in our experiment. It seems that the intersection topic itself makes the VSM unsuccessful. After comparing the assessor topics with the intersection topics, we guess the reason maybe that the natural language style of the assessor topics makes them appropriate to be represented and computed with vectors, while the intersection topics are not, because the different dimensions of an intersection topic vector have no internal relations as organic as those of a natural document. Another reason we guess is that there are few relevant documents on each topic in the testing set that can be used to adjust the profile vector. In TREC-10 our system has proved suitable for “big” topics but not so for “small” topics. The results of last year have also proved that as long as enough relevant documents can be provided, on the intersection-like topics we can still obtain good performance. Although in such circumstances we may not make a good initial profile vector, enough feedback can greatly adapt it to the best position. But this year the case is different. We don’t have so many relevant documents, so the weakness of VSM on the intersection topics becomes distinct. An evidence is that our system still gets better scores on most intersection topics with relative more relevant documents, such as topic R164, R175, R185, R186 and R199.

In next step, our goal is to find a new way to effectively process the semi-automatically made intersection topics. We believe such topics represent the trend in future and are worthy of much more efforts. Accomplishment of the efforts will to some extent lighten assessors` burden in the filtering task.

2.2 Batch Filtering and Routing Subtasks

2.2.1 Text Representation

In our batch and routing filtering system, when preprocessing the documents, we give additional prominence to the words that occur in the *<title>* field and we only use *TF* weight in the vector representation.

2.2.2 Samples Preprocessing

We believe some samples in the training set are not good enough to train the classifier, so we want to eliminate them beforehand. Indeed, samples have different weights since features of documents have different weights. Importance of samples and importance of features are closely related:

- An important sample contains many important features;
- An important feature appears in many important samples;

We calculate the weights of samples as following:

Let A_{mn} is the matrix of the feature frequency in each sample, m is the number of the documents and n the number of the features. a_{ij} is the frequency of the jth feature in the ith sample.

The weight vectors of samples and features are respectively $W_f = (W_{f_1}, W_{f_2}, \cdots, W_{f_m})'$

$W_t = (W_{t_1}, W_{t_2}, \cdots, W_{t_n})'$. Their initial values are $W_f^{(0)}$ and $W_t^{(0)}$, with each component set to 1.

The formulas below are to compute the weights. It can be proved that the computing process is convergent.

$$W_{t_j}^{(k+1)} = \sum_{i=1}^{m} A_{ij} * W_{f_i}^{(k)} \qquad (2.4)$$

$$W_{f_i}^{(k+1)} = \sum_{j=1}^{n} A_{ij} * W_{t_j}^{(k+1)} \qquad (2.5)$$

$$(j = 1, 2, \ldots, n, \quad i = 1, 2, \ldots, m)$$

After computing the weights of all samples, for each topic, we remove the lowest 10% samples and use the remaining samples to train the classifier.

2.2.3 Training

The system uses Rocchio method in the training process. For topic *i*, its representative feature vector $\vec{P}_i$ is calculated as following:

$$\vec{P}_i = \vec{P}_+ - \beta \vec{P}_- \qquad (2.6)$$

Where $\vec{P}_+$ is the centroid of the relevant documents and $\vec{P}_-$ is the centroid of the irrelevant documents in the training set, β is an experiential parameter.

Since the file *filter2002_qrels.test* cannot be used for training, we use the training set to choose proper values of β and the threshold by LOOCV (*Leave-one-out cross-validation*), which is the most extreme and most accurate version of cross-validation.

In test process, those documents with high *cosine* distance to $\vec{P}_i$ are retrieved to form the final results.

2.2.4 Evaluation Results and Analysis

We have submitted two batch-filtering runs and two routing runs. All of them are optimized for T11U. The only difference between the two runs are thresholds and the parameter β in the formula (2.6).

Batch Filtering

The evaluations of batch results are shown in Table 2.4. Table 2.4 shows that in each run, the scores of T11U and T11F are close to medians. For the first 50 topics, we get a set precision higher than the median, but the set recall is lower than it. For the last 50 topics, we set a very strict threshold to avoid T11U becoming negative, because the baseline of T11U is 0.333. As a result, the scores of T11F, Set Precision, and Set Recall are all very low. Since we have set the same threshold for all 100 topics, we think the results show that the threshold for every topic should be different.

Run ID	T11U	Median T11U	T11U vs. median (Topic nums)			T11F	Median T11F	T11F vs. median (Topic nums)		
			>	=	<			>	=	<
ICTBatFT11Ua(1-50)	0.35	0.377	20	10	20	0.18	0.233	19	5	26
ICTBatFT11Ub(1-50)	0.323		15	7	28	0.248		26	7	17
ICTBatFT11Ua(51-100)	0.333	0.254	47	2	1	0	0.024	0	17	33
ICTBatFT11Ub(51-100)	0.304		40	4	6	0.011		4	17	29

Table 2.4 ICT batch filtering runs (all 100 topics) in TREC-11

Routing

We set a lower threshold to get 1000 documents for each topic to form the routing results. The only difference between the two runs is the parameter β.

Run ID	Average precision	Average precision vs. medians(Topic nums)			All Results		
		>	=	<	min	med	max
ICTRouFT11Ua(1-50)	0.243	26	7	17	0	0.223	0.507
ICTRouFT11Ub(1-50)	0.25	31	8	11			
ICTRouFT11Ua(51-100)	0.024	18	16	16	0	0.02	0.085
ICTRouFT11Ub(51-100)	0.025	18	18	14			

Table 2.5 ICT routing runs (all 100 topics) in TREC-11

We can see that all of our results are similar to the medians. We think this is because we only set one same threshold for all topics and lack an effective parameter optimization method. We will try to research on automatic parameter optimization methods.

In the future, we have a lot of work to do to improve our work. For feature selection, we want to use N-Gram to add more terms to represent the documents. For the last 50 topics, we have tried to use KNN to improve the classification results. To our surprise, its result is much worse than the Rocchio method. We will research on the phenomenon and try more complex methods.

3. Web Track

3.1 Introduction

Last year we took part in TREC for the first time and we only submitted four runs for the ad hoc task. This year we submitted runs for both two tasks.

This year, Web track consists of two new subtasks: the Named Page Finding task, which is introduced to investigate methods for finding a particular page that has been named by the user, and the Topic Distillation task, which is introduced to investigate methods for finding key resources in a particular topic area. In the former task, the system should return a single named page as the result. For instance, for the query *"passport application form"*, the correct answer should be the page *travel.state.gov/dsp11.pdf*, which contains the electronic copy of requested form. In the Topic Distillation task, a single relevant document is not important any more. The concept *resource* is introduced as the basic element of results and judgments. The test collection of this year's Web track is changed to *.Gov* data set which substitutes *Wt10g* used in previous years.

Though the Web track tasks have been significantly modified, the basis of experiments is still the traditional IR systems. In TREC 2001 we investigated the effectiveness of the combination of classical Boolean model and probabilistic model in the ad hoc task. We also investigated methods that make use of link information between pages in the same task. Neither of the results was as good as we had expected. So this year we decide to adopt vector space model and to make use of only text contents and internal structure of pages. Our retrieval system is based on SMART. In order to deal with large data set such as *Wt10g* and *.Gov* test collection, we modified the basic SMART system, and the *Lnu-Ltu* weighting method was added to the system. This method has been proven to be very effective and efficient in our experiments. The classical weighting methods such as *lnc-ltc* do not behave well in our experiments.

3.2 Named Page Finding Task

As introduced above, the goal of Named Page Finding task is to find appropriate page(s) named by users. It is rather close to a special kind of user requirement, i.e., finding a few documents that precisely meet the information need of users. The query *"passport application form"* is an example. Another one is the query "table of contents gnu make manual", by which a user would like to find the exact page that is the table of contents of GNU make manual. By analyzing these examples we have found some features that can be utilized.

Firstly, the content-based ranking score of traditional IR system is still the most important factor in Named Page Finding. If we assign the content-based score a less important coefficient in result merging process that will be described below, the final results will be worse. This can be explained if we notice that single term is more important in Named Page Finding task than in ad hoc task. This task pays more attention to precision than to recall. Only those pages that contain all or most of the query terms would have high possibility of meeting information need implied by the query in, thus they would have higher content-based scores than most of the irrelevant documents. Certainly some of irrelevant documents will also have high content-based scores, but we will enhance the scores of relevant documents by result merging process.

Secondly, the internal structure of documents will give us plenty of information. As the name of task suggests, query terms of Named Page Finding task are the names of relevant documents. Usually they are precise representations of topics. They should more possibly appear in important positions such as document title, beginning sentences of paragraphs and section headers, or display in a striking manner, for example a bold, italic, and large size font face. In such situation authors of documents have explicitly defined them as important terms. We can get a lot of relevance information by comparing the query terms with them. Besides this there is another reason why the method is especially useful for the task. Queries in ad hoc task are often about general topics. They must be described by natural language so that people can understand the information need under which the queries are developed. So they are prone to ambiguity. Correspondingly the relevant documents cannot be named clearly and easily. On the contrary, the information need of Named Page Finding task can be very easily understood, even without extra descriptions, so authors and searchers of the same documents will in the gross adopt the same terms as topic descriptions. The Homepage Finding task in last year's web track can be regarded as a kind of Named Page Finding task. In fact, when we added the phrase "home page" to the original queries we got obvious improved results. In our contrast experiment, ad hoc runs using document structure information gave poor results whose average precisions are too low to be mentioned while Homepage Finding runs gave fairly satisfactory results.

The last factor we have proven to be effective for the Named Page Finding task is anchor texts of documents. They act as almost the same role as the second factor. They can be regarded as names given by referrers to target documents. When the target documents can be easily named and referrers adopt the same names widely, retrieval results using the names are fairly satisfactory.

As we have stated above we believe that Homepage Finding task is a special kind of Named Page Finding task. So except some special methods for Homepage Finding such as analysis of URL depth, the methods that are effective for Homepage Finding should also be effective for Named Page Finding. We ran our experiments on *Wt10g* data set using topics and qrels developed for the Homepage Finding task to find the most optimized parameters. The results are shown in Table 3.1 and Table 3.2. We then applied the same system to the *.Gov* data set and Named Page Finding task. The experimental results that we observed have proven to be satisfactory.

We use the linear result merging method to get the last result of Named Page Finding task. The merging formula is

$$W(p) = \alpha * w_c(p) + \beta * w_s(p) + \gamma * w_a(p) \qquad (3.1)$$

Where $w_c(p)$ is the content weight of page p, $w_s(p)$ is the weight from structure information, $w_a(p)$ is the weight from the anchor text of page p and α, β, γ are their coefficients. In our experiments only the titles of documents are used as structure information. The evaluation results are shown in Table 3.3.

Average Precision	R-precision	Recall
0.1938	0.2185	2243

Table 3.1 Our content-based experiment for the ad hoc task of TREC-10.

Content(α)	Structure(β)	Anchor text(γ)	MRR	Correct Answers
1	0	0	0.4185	122/145
0	1	0	0.4467	105/145
0	0	1	0.3769	94/145
1	0.5	0.5	0.5880	133/145
1	0.5	0.8	0.6032	130/145
1	0.5	1	0.5806	130/145

Table 3.2 Our Homepage Finding experiments of TREC-10

Run ID	MRR	Answers Found@10	Not Found@all
ictnp2	0.559	114/150	18/150
ictnp3	0.557	116/150	18/150
ictnp4	0.555	116/150	18/150
ictnp6	0.613	127/150	14/150
ictnp7	0.613	127/150	14/150

Table 3.3 ICT Named Page Finding runs in TREC-11

3.3 Topic Distillation Task

As described in the TREC-2002 Web Track Guideline, a key resource might be:

- The home page of a site dedicated to the topic.
- The main page of a sub-site (part of a site) dedicated to the topic. (If there are several relevant pages but no main page linking them, then the individual pages must be judged on their own merit.)
- A highly useful html, pdf, doc, ps page dedicated to the topic (should be an outstandingly useful page). Return the page's URL.
- A highly useful page of links (hub page) on the topic. Return its URL.
- A relevant service e.g. perhaps http://www.nasa.gov/search/ for the NASA topic.

Except the last two cases key resources are some important pages inside individual sites. Our first experiment was based on HITS algorithm. We submitted queries to SMART and retrieved ranked page lists, and then applied HITS to every group of pages coming from the same site. We extracted the page that had the maximum Hub+Authority value from each group of pages and added them to the final result. We found that the average result of this method was disappointing, partly because many Hub and Authority pages computed by HITS cannot meet the definition of key resource. Our last experiment on this task was based on a simple idea. After the first retrieval, we scanned the page list. If we found a page's url containing the other's, we then re-weighted the latter page by adding the former's weight to the latter's. After re-weighting the weight of a certain result page x is

$$w = \sum_{p} \frac{w_p}{\sqrt{r_p}} \qquad (3.2)$$

Where p is a page whose url string contains x's. w_p is the content weight of page p and r_p is the rank of page p. The run icttd2 is based on this approach, and ictted3 is based on icttd2 plus some additional re-weighting methods. The evaluation result is shown in Table 3.4.

The run icttd1 is a baseline run produced by our retrieval system. It is the best one among the three runs. It seems that our re-weighting methods are not so effective as we have expected. We believe that more attentions should be paid to the instances of key resources given by the TREC qrels so that characters of them can be found.

Run ID	Average Precision	R-Precision	Rel_ret
icttd1	0.1620	0.1919	1038/1574
icttd2	0.1364	0.1599	1038/1574
icttd3	0.0597	0.1034	288/1574

Table 3.4 Result of Topic Distillation task in TREC-11

4. Conclusion

We've participated in the TREC conference for two times. By communicating with the researcher all over the world, we've learned more. We've got many experiences in English information processing, which will benefit us greatly in our Chinese information processing.

TREC not only advances our research on IR, but also enlighten our insights. From here, we can find our advantages and disadvantages comparison to the foreign friends going the same way. We are glad to take part in TREC continuously.

Acknowledgements

This research is supported by the national 973 fundamental research program under contact of G1998030413, the Institute Youth Fund under contact 20016280-9 and the Institute Youth Fund under contact 20026180-24. We give our thanks to all the people who have contributed to this research and development, in particular Yanbo Han, Li Guo, Qun Liu, Xin Zhang, Hao Zhang, Dongbo Bu and Huaping Zhang.

References

[1] Ogawa, Y., Mano, H., Narita, M., Honma, S. Structuring and Expanding Queries in the Probabilistic Model. In *The Ninth Text REtrieval Conference (TREC 9),2000.*

[2] O. Yasushi, M. Hiroko, N. Masumi, H. Sakiko. Structuring and expanding queries in the probabilistic model. In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[3] S.E. Robertson, S. Walker. Okapi/Keenbow at TREC-8. In *The Eighth Text REtrieval Conference (TREC 8),1999.*

[4] S.Brin and L.Page. The anatomy of a large scale hypertextual web search engine. In *The 7th WWW Conference,1998.*

[5] Lawrence Page,Sergey Brin,Rajeev Motwani,Terry Windograd. The Pagerank citation ranking: Bring order to the web. *Stanford Digital Libraries working paper, 1997-0072.*

[6] J.Kleinberg. Authoritative sources in a hyperlinked environment. *Proc 9th ACM-SIAM SODA,1998.*

[7] Ian H. Witten, Alistair Moffat, Timothy C. Bell. *Managing gigabytes: Compressing and indexing documents and images, 2nd ed, 1994.*

[8] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford. OKAPI at TREC-3, In *The Third Text REtrieval Conference (TREC 3),1994.*

[9] S. Robertson, I. Soboroff, The TREC 2001 Filtering Track Report, In *The Tenth Text REtrieval Conference (TREC 10)*, page 26, 2001.

[10] B. Wang, H. Xu, Z. Yang, Y. Liu, X. Cheng, D. Bu, S. Bai, TREC-10 Experiments at CAS-ICT: Filtering, Web and QA, In *The Tenth Text REtrieval Conference (TREC 10)*, page 109, 2001.

[11] Yi Zhang, James P. Callan, Maximum Likelihood Estimation for Filtering Thresholds, SIGIR 2001, page 294-302, 2001.

[12] Y. Zhang, J. Callan, The Bias Problem and Language Models in Adaptive Filtering, In *The Tenth Text REtrieval Conference (TREC 10)*, page 78, 2001.

[13] T. Ault, Y. Yang, kNN, Rocchio and Metrics for Information Filtering at TREC-10, In *The Tenth Text REtrieval Conference (TREC 10)*, page 84, 2001.

[14] S. Alpha, P. Dixon, C. Liao, C. Yang, Oracle at TREC 10: Filtering and Question-Answering. In *The Tenth Text REtrieval Conference (TREC 10)*, page 423, 2001.

[15] Rocchio, J. J. Relevance Feedback in Information Retrieval. In *The SMART Retrieval system, Prentice-Hall, Englewood NJ. 1971, 232-241.*

PLIERS AT TREC 2002

A MacFarlane
Centre for Interactive Systems Research, Department of Information Science,
City University, Northampton Square, LONDON EC1V 0HB, UK

Abstract: We describe our experiments with the .GOV collection in both the topic distillation and named page tasks at the 2002 TREC web track. We report on our indexing speed, retrieval efficiency results and effectiveness results for both tasks.

1. Introduction

We report on our experiments for the TREC 2002 web track for both the topic distillation and named page tasks. We use a very simple method for both tasks which takes the first hit page in the top 10 for a give web site and discards any further pages from that web site (section 2 describes our research aims and objectives in more detail). We also describe indexing results (section 3), give a description of the runs and settings used (section 4), briefly describe our retrieval efficiency results in section 5, and outline our retrieval efficiency results in sections 6 and 7. A conclusion is given in section 8.

2. Research aims and objectives

We take a very simple approach both the topic distillation and named page tasks. We want to test the hypothesis "does the best BM25 ranked document from any given web site yield the best web page for users information needs". We want to compare this rather simple technique with other more complex techniques which use link information in order to find the best given web page or pages.

Our retrieval efficiency experiments differ from our previous work [2] which concentrated on using large scale parallelism to speed up the processing of both indexing and search. In these experiments we want to show that we can successfully process large amounts of text with our system using a single machine (even if it does has multiple processors on it).

3. Indexing methodology and results

3.1 Indexing methodology

We used a simple and straightforward methodology for indexing: parsing, remove stop words, stemming in the given language. The PLIERS HTML/SGML parser needed to be altered to detect non-ASCII characters such as those with umlauts, accents, circumflexes etc. We also incorporated non-English stemmers into the PLIERS library (these were not used for these experiments). We used a standard stop word list defined by Fox [3]. Apart from this our indexing methodology is much the same a described in [2].

3.2 Indexing results

Elapsed Time (hrs)	Dictionary file size MB	Postings file size GB	Map file size MB	% of text
10.54	110	1.17	40.4	7%

Table 1 – Indexing results for .GOV collection

Table 1 gives the indexing results for the .GOV collection. PLIERS was able to process the data in a reasonable time (just under 11 hours) and produced an inverted file that was only 7% of the collection size. This compares favourably with our previous web track experiments with WT100g [2], in which indexes were 11% of the collection size. The final merge took only about 10 minutes (a total of 1.5% of total indexing time): this represents a significant improvement on previous single processor experiments. This can be explained by our usage of a significantly faster machine. We regard it as a success to be able to index data of this size: we suspect that the system would not be able to handle a slightly larger collection without failing.

4. Run descriptions and settings used

All experiments were conducted on a Pentium 4 machine with 256 MB of memory and 240 GB of disk space. The operating system used was Red Hat Linux 7.2. All search runs were done using the Robertson/Sparck Jones Probabilistic model. All our runs are in the Web track. All queries derived from topics are automatic.

Changes to software in order to conduct these particular experiments were minimal. We used the URL/TREC ID list supplied with the .GOV collection to identify and eliminate documents from the top 10 results which are from the same web site. Only the highest ranked document from a web site is retained. The top 10 results are therefore guaranteed to have unique URL's in them i.e. all documents in the top 10 are from different web sites. We used this technique on both Web track tasks.

The weighting function used for these experiments was BM25 [1]. There are a number of tuning constants for this function with which we have done experiments on before, in order to find the best combination for search [2]. There are two constants: **K1** and **B** [1]. The **K1** constant alters the influence of term frequency in the BM25 function, while the **B** constant alters the influence of normalised average document length. Values of **K1** can range from 0 to infinity, whereas the values of **B** are with the range 1 (document lengths used unaltered) to 0 (document length data not used at all). Table 2 shows the details of our official Web track runs [Note: T = Title only queries, TD=Title and Description, D=Description only].

Run ID	**Description**	**Query Type**	**K1 Constant**	**B Constant**
pltr02wt1	Distillation run	T	1.5	0.8
pltr02wt2	Non-Distillation run	T	1.5	0.8
pltr02wt3	Distillation run	T	1.5	0.2
pltr02wt4	Non-Distillation run	T	1.5	0.2
pltr02wt5	Distillation run	TD	1.5	0.2
pltr02wt6	Named page run	D	1.5	0.2
pltr02wt7	Named page run	D	1.5	0.4
pltr02wt8	Named page run	D	1.5	0.6
pltr02wt8	Named page run	D	1.5	0.8

Table 2 – TREC 2002 Web track run details

We used 1.5 for the **K1** constants for all our runs as this was the best found in our previous Web track experiments for a large collection of web data [2]. For the topic distillation task we varied the **B** constant between 0.2 and 0.8 in order to investigate the effect of document length on this task. We also included some non-distillation runs to allow us to quantify the effectiveness of our distillation runs. Most of our distillation task runs used title only queries (realistic), but we did submit one title/description run. We used description only queries for the named page task (this was the only allowed method). We were able to vary the **B** constant on the named page task a little more as we had less flexibility on those runs: this allowed us to investigate the effect of document length in more detail for this task.

5. TREC 2002 retrieval efficiency results

5.1 Retrieval efficiency results

Table 3 gives a sample of the average elapsed time for each of the official runs. The distillation task runs contained 50 queries, whilst the named page runs contained 150 queries. We are very satisfied with our query response times on the .GOV collection. All our runs have met the one to ten second response time criteria specified by Frakes [5], and they are good for a collection of this size. We believe that these response times could be considerably improved by using various query optimisation techniques (currently we do not use any in our query processing).

Query Type	Distillation runs	Non-Distillation runs	Named page runs
T	1.24	1.29	-
TD	7.17	-	-
D	-	-	1.48

Table 3 – TREC 2002 average elapsed time for official runs (sample)

6. Topic distillation task results

The topic distillation results are shown in Table 4.

Run ID	Description	Precision @ 10	Average Precision	Query Type	B
pltr02wt1	Distillation run	0.200	0.144	T	0.8
pltr02wt2	Non-Distillation run	0.241	0.190	T	0.8
pltr02wt3	Distillation run	0.175	0.109	T	0.2
pltr02wt4	Non-Distillation run	0.200	0.143	T	0.2
pltr02wt5	Distillation run	0.088	0.044	TD	0.2

Table 4 – TREC 2002 Topic distillation results

An interesting result from our experiments was that the Non-distillation runs did better than the Distillation runs, and that one of our Non-distillation runs (pltr02wt2) came second overall in this years Web track Topic distillation task [5]. Two significant observations can be made about these experiments. The first is that just using a simple minded URL removal technique to improve topic distillation simply does not work. The second is that for this task, using ordinary BM25 search techniques with no relevant feedback is comparable to those methods which utilize such evidence as document structure, anchor text and link structure. With respect to the BM25 tuning constant parameter it is clear that a lower value of B was better for both our types of runs: runs with **B** set at 0.8 did better than those with **B** set at 0.2 (when comparing like with like e.g. distillation runs).

7. Named page task results

The named page results are shown in Table 5.

Run ID	MRR	% in top 10	% not found	B
pltr02wt6	0.334	44.7%	44.0%	0.2
pltr02wt7	0.414	53.7%	41.3%	0.4
pltr02wt8	0.416	52.7%	41.3%	0.6
pltr02wt8	0.418	52.7%	42.0%	0.8

Table 5 – TREC 2002 Named page task results

Overall the results are disappointing: in most runs we are only finding about 50% of the named pages in the top 10, and our experiments do not find up to 40% of the resources at all. Therefore our MRR results are not as good as we would have liked – up to something in the region of 0.72 as found with the top scoring run in this years Named page task [5]. We believe that one important factor may be the cause of reduced effectiveness for this task given the evidence found in topic distillation runs: all experiments used the URL removal technique – and this has obviously had a significant effect on our MRR scores. It would be useful to do Named page experiments without the URL removal procedure in order to quantify the effect of using such a method. We could also make a contribution to the IR community, being able to compare a realistic BM25 technique with those which make use of document/link structures and anchor text. It should be noted that MRR increases with the value of B, but the increase is not significant beyond B=0.4. The increase from B=0.2 to B=0.4 is significant however: the percentage increase is 24%. Increases on the other runs with increasing value of B are all below the half percent mark.

8. Conclusion

The simple minded technique of removing multiple hits from web pages used for the purposes of the experiments described in this paper, do not appear to have work particularly well. We have found, significantly, that a straight BM25 term weighting run with no relevance feedback compares very well indeed with methods which use document/link structures and anchor text in the Topic distillation task. Our Named page runs are disappointing, and we believe that part of the problem relates to removing multiple hits from web pages.

With respect to our hypothesis, we have demonstrated that for the topic distillation task, BM25 appears to work quite well. However we have not been able to demonstrate this for the named page task and further investigation is required. In particular the issues of removing documents from the top 10 when other document from the same web site have already been retrieved needs to be investigated.

The evidence from the experiments described in this paper show that altering the value of the B constant in the BM25 model does appear to have an effect: in particular a high value of B parameter appears to work well with the .GOV collection for both of this years web track tasks. We have been able to show that our system scales to much larger collections of the .GOV size, and have shown the indexing/search speeds are acceptable for data sets of this size.

References

[1] Robertson, S.E., and Sparck Jones, K., Simple, proven approaches to text retrieval, University of Cambridge Technical report, May 1997, TR356 ,
[http://www.cl.cam.ac.uk/Research/Reports/TR356-ksj-approaches-to-text-retrieval.html] – visited 22nd July 2002.

[2] MacFarlane, A., Robertson, S.E.., McCann, J.A., PLIERS AT TREC8, In: Voorhees, E.M., and Harman, D.K., (eds), The Eighth Text Retrieval Conference (TREC-8), NIST Special Publication 500-246, NIST: Gaithersburg, 2000, p241-252.

[3] Fox, C., A stop list for general text, SIGIR FORUM, ACM Press, Vol. 24, No. 4. December 1990.

[4] Frakes, W.B., Introduction to information storage and retrieval systems. In: Frakes, W.B. and Baeza-Yates, R. (eds), Information retrieval; data structures and algorithms, Prentice Hall, 1992, p1-12.

[5] Craswell, N., and Hawking, D., Overview of the TREC-2002 Web Track, [in this volume].

Question Answering Using XML-Tagged Documents

Kenneth C. Litkowski
CL Research
9208 Gue Road
Damascus, MD 20872
ken@clres.com

Abstract

The official submission for CL Research's question-answering system (DIMAP-QA) for TREC-11 only slightly extends its semantic relation triple (logical form) technology in which documents are fully parsed and databases built around discourse entities. We were unable to complete the planned revision of our system based on a fuller discourse analysis of the texts. We have since implemented many of these changes and can now report preliminary and encouraging results of basing our system on XML markup of texts with syntactic and semantic attributes and use of XML stylesheet functionality (specifically, XPath expressions) to answer questions.

The official confidence-weighted score for the main TREC-11 QA task was 0.049, based on processing 20 of the top 50 documents provided by NIST. Our estimated mean reciprocal rank was 0.128 for the exact answers and 0.227 for sentence answers, comparable to our results from previous years. With our revised XML-based system, using a 20 percent sample of the TREC questions, we have an estimated confidence-weighted score of 0.869 and mean reciprocal rank of 0.828. We describe our system and examine the results from XML tagging in terms of question-answering and other applications such as information extraction, text summarization, novelty studies, and investigation of linguistic phenomena.

1 Introduction

In previous years, DIMAP-QA was based on full parsing of the 10 or 20 NIST-supplied top documents of TREC texts and extracting semantic relation triples into a database from which answers were then found (Litkowski, 2001; Litkowski, 2002a). As noted previously, our system was intended to be part of a larger system of discourse analysis of the texts, which had not been sufficiently implemented to serve as the basis for question-answering. In addition, although our idea of capturing semantic relation triples (identifying discourse entities and their relations to other discourse elements) seemed sound, the use of a traditional database structure made it difficult to represent and exploit the structural properties of natural language.

Extensible Markup Language (XML) provides a more natural mechanism for representing texts. A valid XML document is a tree and we can readily design our entire representation on this tree structure. The entire TREC collection (or any subset of documents) can be represented as one tree; the next level of the tree represents each document. At the next level, each document may be represented as a set of sentences, each of which may then be subdivided into sentence segments or clauses (elementary discourse units), which are then broken down into traditional parse trees, ending in leaf nodes corresponding to the words in the sentences. Each node in the tree may have associated attribute names and values.

A key part of the XML design philosophy is the ability to transform an XML file into usable output for display or other purposes (e.g., populating a database). This is accomplished via XML stylesheet language transformations (XSLT). XSLT is based on the creation of XPath expressions, which specify the path from the top of the XML tree to some intermediate or leaf node. For question-answering, an XPath expression (query) essentially specifies search criteria that return a string answer, based on node types and attributes. For example, for question 1553 ("who makes Magic Chef refrigerators?"), a single XPath expression looks for a sentence containing "Magic Chef" and an anaphor with an antecedent, finds the sentence containing the antecedent, notices that the antecedent is a predicate to "is", and retrieves the subject of the predicative, "Maytag", as the answer to the question.

Section 2 presents the TREC QA problem description. Section 3 describes our system: sentence splitting, parsing, discourse and sentence analysis, and database development and XML tagging. Section 4 briefly describes question-answering against the document databases. Section 5 provides a detailed description of procedures used to answer questions from XML-tagged documents. Section 6 presents and analyzes our official results and the unofficial results achieved using the XSLT approach. Section 7 describes anticipated next steps for improving the question-answering capability and for using XML-tagged documents in other applications such as information extraction, text summarization, novelty studies, and investigation of linguistic phenomena.

2 Problem Description

Participants in the main TREC-11 QA track were provided with 500 unseen questions to be answered from the AQUAINT Corpus of English News Text on two CD-ROMs, (about one million documents), containing documents from *Associated Press Newswire*, *New York Times Newswire*, and *Xinhua News Agency*. These documents were stored with SGML formatting tags (XML compliant). Participants were given the option of using their own search engine or of using the results of a "generic" search engine. CL Research chose the latter, relying on the top 50 documents retrieved by the search engine. These top documents were provided simultaneously with the questions.

Participants in the main task were required to answer the 500 questions with a single exact answer, containing no extraneous information and supported by a document in the corpus. A valid answer could be NIL, indicating that there was no answer in the document set; NIST included 46 questions for which no answer exists in the collection. Answers for the 500 questions were to be sorted according to a participant's confidence. NIST evaluators next judged whether an answer was correct, inexact, unsupported, or incorrect. The submissions were then scored as the sum with I from 1 to 500 of the number of correct answers up to I divided by I, and this sum divided by 500, called a confidence-weighted score (CWS).

CL Research performed two runs for the main task. However, we mistakenly submitted the second run, based on using the top 20 documents (rather than one for the top 10 and one for the top 20), for both run tags. The discussion of our official submission will thus present results for only one run.

3 System Description

The CL Research question-answering system consists of four major components: (1) a sentence splitter that separated the source documents into individual sentences; (2) a parser which took each sentence and parsed it. resulting in a parse tree containing the constituents of the sentence; (3) a parse tree analyzer that identified important elements of the sentence and created semantic relation triples stored in a database and a set of discourse constituents (sentences and clauses, discourse entities, verbs and prepositions) used to create an XML-tagged version of each document; and (4) two question-answering programs, one using the database and one using the XML documents.

3.1 Sentence Splitting

Sentence splitting proceeded as described in previous years (Litkowski, 2002a; Litkowski, 2001). Using the new AQUAINT collection posed some difficulties because of idiosyncratic markup (described below). Unlike previous years, this phase of processing was completely robust.

For TREC-11, the top 20 documents (as ranked by the search engine) were analyzed for the main task, with one database containing only the processing for the top 10 documents and the other for the full 20 documents. Overall, this resulted in processing 10,000 documents from which 257,276 sentences were identified and presented to the parser. Thus, we used an average of 25.7 sentences per document (compared to 22.8 in TREC-10, 28.9 in TREC-9 and 31.9 in TREC-8) or 257 sentences for the 10-document set and 514 for the 20-document set.

3.2 Parser

We continued our use of the Proximity parser, described in more detail in our previous papers (Litkowski, 2002a; Litkowski, 2001). As described there, the parser output consists of bracketed parse trees, with leaf nodes describing the part of speech and lexical entry for each sentence word. Annotations, such as number and tense information, may be included at any node. Usable output was generated by the parser for 99.9 percent of the sentences that were processed.

3.3 Discourse and Sentence Analysis

The sentence parsing in the CL Research system is part of a broader system designed to provide a discourse analysis of an entire text or set of texts. We are using this system for processing encyclopedia articles, historical texts, scientific articles[1], as well as the news wire texts in TREC and the RST treebank (Linguistic Data Consortium, 2002). Frequently, the

[1] See http://www.clres.com/sa-articles.xml.

input has already been tagged (e.g., in SGML) and our processing may result in additional tagging.

After each sentence is identified and parsed, its parse tree is traversed in a depth-first recursive function. During this traversal, each non-terminal and terminal node is analyzed, making use of parse tree annotations and other functions and lexical resources that provide "semantic" interpretations of syntactic properties and lexical information.

At the top node in the tree, just prior to iteration over its immediate children, the principal discourse analysis steps are performed. Each sentence is treated as an "event" and added to a list of events that constitute the discourse. We first update data structures used for anaphora resolution. Next, we perform a quick traversal of the parse tree to identify discourse markers (e.g., subordinating conjunctions, relative clause boundaries, and discourse punctuation) and break the sentence down into elementary discourse units. We also identify and maintain a list of the sentence's verbs at this stage, to serve as the bearers of the event for each discourse unit.

After the initial discourse analysis, the focal points in the traversal of the parse tree are the noun phrases. When a noun phrase is encountered, its constituents are examined and its relationship to other sentence constituents are determined. The relationship analysis gives rise to a **semantic relation triple**, which consists of a discourse entity (the noun phrase itself), a syntactic or semantic relation which characterizes the entity's role in the sentence, and a governing word to which the entity stands in the semantic relation. A triple is generally equivalent to a logical form (where the operator is the semantic relation) or a conceptual graph, except that a semantic relation is not strictly required. with the driving force being the discourse entity.

Each noun phrase is added to a list of discourse entities for the entire text, that is, a "history" list. As each noun phrase is encountered, it is compared to discourse entities already on the history list. This comparison first looks for a prior mention, in whole or in part, to determine whether the new entity is a coreferent of a previous entity (particularly valuable for named entities). If the new entity is an anaphor, an anaphoric resolution module is invoked to establish the antecedent. A similar effort is made to find antecedents for definite noun phrases. The noun phrase's constituents are examined for numbers, adjective sequences, possessives (which are also subjected to the anaphoric resolution module), genitive determiners (which are made into separate discourse entities), leading noun sequences, ordinals, and time phrases. Finally, an attempt is made to assign a semantic type to the head noun of the phrase using WordNet or an integrated machine-readable dictionary or thesaurus.

If a noun phrase is part of a prepositional phrase, a special preposition dictionary is invoked in an attempt to disambiguate the preposition and identify its semantic type. This module identifies the attachment point of the preposition and uses information about the syntactic and semantic characteristics of the attachment point and the prepositional object for this disambiguation. The preposition "definitions" in this dictionary are actually function calls that check for such things as literals and hypernymy relations in WordNet. A list of all prepositions encountered in the text is maintained as the text is processed. (See Litkowski (2002b) for further details.)

Predicative adjective phrases, relative clauses, subordinate clauses. and appositives are also flagged as the parse tree is traversed. The attachment points and spans of relative clauses and appositives are noted.

As the noun phrases are encountered, we attempt to identify the syntactic or semantic role they play in the sentence. These include "SUBJ," "OBJ", "TIME," "NUM," "ADJMOD," and the prepositions heading prepositional phrases. Relative clauses and appositives are inherently modifiers of their attachment points.

The governing word was generally the word in the sentence that the discourse entity stood in relation to. For "SUBJ," "OBJ," and "TIME," this was generally the main verb of the sentence. For prepositions, the governing word was generally the noun or verb that the prepositional phrase modified. (Because of the context-sensitive dynamic parsing goals that were added when a verb or a governing noun was recognized, it was possible to identify what was modified.) For the adjectives and numbers, the governing word was generally the noun that was modified.

A semantic relation and a governing word were not identified for all discourse entities. Notwithstanding, a list of every discourse entity is maintained with a unique identifier and all characteristics that can be associated with them.

3.4 Database Development and XML Tagging

The text analysis module generates two types of output: (1) a database of semantic relation triples and (2) an XML tagging of the text. Either type of output is optional. For the database, each semantic relation triple is added as it is generated. Overall. 2,306,698 semantic relation triples were created in parsing the

257,276 sentences, an average of 9.0 triples per sentence (compared to 9.7 in TREC-10).

Although we have achieved some degree of success with the database approach, we have found that it is difficult to work with. The table of semantic relation triples is not intuitive because the flat structure removes the tree structure that is inherent in a grammar-based parser. For simple questions, answering is a matter of forming a join between a question database parsed and analyzed in the same way as the document database. With greater complexity, and with a document database where a simple join does not produce an answer, the logic required to examine a path of relations becomes more difficult.

As indicated above, the text analysis module develops four lists at the same time as the semantic relation triples: (1) events (the discourse segments), (2) entities (the discourse entities), (3) verbs, and (3) semantic relations (the prepositions). Each document consists of one or more tagged segments, which may include nested segments. Each discourse entity, verb, and preposition in each segment is then tagged. A segment may also contain untagged text, such as adverbs and punctuation. Each item on each list has an identification number (used in many of the functions of the text analysis module). As indicated above, the discourse analysis assigns attributes to each segment (and subsegment), discourse entity, verb, and preposition.

For segments, the attributes include the sentence number (if the segment is the full sentence), a list of subsegments (if any), the parent segment (if a subsegment), the text of the segment, the discourse markers in the sentence, and a type (e.g., a "definition" sentence or "appositive"). For discourse entities, the attributes include its segment, position in the sentence, syntactic role (subject, object, prepositional object), syntactic characteristics (number, gender, and person), type (anaphor, definite or indefinite), semantic type (such as person, location, or organization), coreferent (if it appeared earlier in the document), whether the noun phrase includes a number or an ordinal, antecedent (for definite noun phrases and anaphors), and a tag indicating the type of question it may answer (such as who, when, where, how many, and how much). For verbs, the attributes include its segment, position in the sentence, the subcategorization type (from a set of 30 types), its arguments, its base form (when inflected), and its grammatical role (when used as an adjective). For prepositions, the attributes include its segment, the type of semantic relation it instantiates (based on disambiguation of the preposition) and its arguments (both the prepositional object and the attachment point of the prepositional phrase).

After all sentences in a document have been processed, the four lists are used to create an XML-tagged version of the document. The XML tagging is performed for each segment within the XML element **segment**, with the attributes listed in the tag opening. The tag content is initialized to the segment text and we proceed to mark up this text according to the text contained within each subsegment, discourse entity (**discent**), verb (**verb**), and preposition (**semrel**) in the segment. As these XML elements are generated, their attributes are added to the tag opening.

The resultant XML-tagged text for individual documents were combined into one overall file of documents, each with a tag for the document number. For TREC, the output consisted of groups of ten documents from the NIST-provided top documents for each question. Since we only processed the top 20 documents, we had 500 XML files for the top ten documents and 500 for documents ranked 11th through 20th. These are the files used for answering the TREC questions.

4 Question-Answering Using Document Databases

For TREC-11, the question-answering against the document databases was little changed from previous years. We refer to our earlier detailed descriptions (Litkowski, 2002a; Litkowski, 2001) and provide only a brief overview here.

For TREC-11, a database of documents was created for each question, as provided by the NIST generic search engine. A single database was created for each question in the main task. The question-answering consisted of matching the database records for an individual question against the database of documents for that question.

The question-answering phase consists of four main steps: (1) detailed analysis of the question to set the stage for detailed analysis of the sentences according to the type of question, (2) coarse filtering of the records in the database to select potential sentences, (3) extracting possible short answers from the sentences, with some adjustments to the score, based on matches between the question and sentence database records and the short answers that have been extracted and (4) making a final evaluation of the match between the question's key elements and the short answers to arrive at a final score for the sentence. The sentences and short answers were then ordered by decreasing score. The short answer for each question (an "exact" answer), its score, and its sentence (the "justification") were printed to a file. This file was then sorted by score to create a "confidence-ordered" answer set submitted to NIST.

5 Question-Answering Using XML-Tagged Documents

As described earlier, question-answering against XML files essentially involves describing a path (XPath) from the top of the tree(s) to a discourse entity (in our case, to a **discent** node) which is returned as the answer. To do this, a question is converted into an XPath expression used to select nodes in the files. For example, for question 1593 ("What percent of Egypt's population lives in Cairo?"), an XPath expression is

```
//segment[contains(.,'Cairo')]
//discent[contains(.,'percent') and
@tag='howmany']
```

The first double slash says to find any node in all documents being searched that are marked as **segment** elements and contains the word "Cairo". The second double slash says to find all **discent** elements that are descendants of such segments containing the word "percent" and that have an attribute **tag** with value equal to "howmany". This XPath expression will return zero or more nodes from however many documents are processed.

In general, question-answering consists of the following steps: (1) analyze the question and convert it into an XPath expression; (2) load the XML file(s) and select the nodes satisfying the XPath expression; and (3) if necessary, score and/or evaluate the nodes returned and present them to the user. The second step is the easiest, consisting of a loop over the files being processed, with a single statement to load the file and another single statement to select the nodes.

The first step, determining the XPath expression, is more difficult. As can be seen for q1593, not all the question elements are present in the query. This may be characterized as a "backoff" strategy, beginning with all the terms in the query and removing some that are not necessary or are too restrictive. For q1593, including all the terms will result in zero nodes. This is frequently the case, with questions often providing much more information than is likely to appear in one sentence. The third step, evaluating the nodes selected, is generally not as complicated; a well-formulated XPath expression generally returns only a couple of answers, although there are some question types that require more extensive processing. We will describe our observations about the first and the third steps in more detail below.

As indicated earlier, we were not able to implement our question-answering against the XML-tagged documents for our official submission. Using these documents has required an entirely new conceptual approach, involving the resolution of many intertwined issues. This new approach has been evolving since our submission; many refinements are necessary and many possibilities for making these changes have been emerging.

To begin with, the whole tagging process described in general terms above requires dealing with virtually the full panoply of natural language processing, including tokenization, sentence splitting, parsing, word-sense disambiguation, anaphora resolution, and discourse analysis. While we have developed a system that comprehends all these components, many of the components have not yet been implemented to the state of the art. For example, our anaphora resolution module is currently estimated at 55 percent correct, whereas the state of the art has been attaining levels over 80 percent. Also, our typing and characterization of prepositional semantic relations is currently operating at about 20 percent (see Litkowski (2002b) for our lexicographic approach to this problem), so that we have to rely on the preposition itself as the bearer of information about the semantic relation. Further, our discourse structure analysis is an initial implementation, presently handling only appositives and relative clauses.

A second major issue to be faced is the selection of tags and their attribute names and values. This issue involves identifying what information will be useful and then developing techniques for extracting the information, using whatever other resources may be available (such as dictionaries and thesauruses). An important question, given our semantic predilections, is what semantic classes to use for characterizing discourse entities. Another important question is how to group information: what sentence parts should be grouped together and which modifiers should be separated or put into attributes of a discourse entity.

Dealing with these issues (identifying problems with the functioning of our XML output generation and examining representational alternatives) is very complex and requires the development of mechanisms for analyzing them. This has led to two steps in our development cycle: (1) the development of an analysis interface for assessing problems and (2) the use of the TREC questions as guidance for inadequacies in our representations. As will be suggested below, the use of XSLT has demonstrated not only a capability for dealing with these issues, but also provides a strong indication that an XML representation of text will be extremely useful for a wide range of applications, including question-answering.

5.1 Step 1: An XML Analysis Interface

The generation of 1000 XML files each containing 10 TREC documents provides a large amount of data; the XML files are approximately five times the size of the TREC documents. The XML files can be viewed (with retention of the nested structure) in Microsoft's Internet Explorer, but this does not allow any systematic examination of the data. Conventionally, those working with XML files develop XML stylesheets for portraying the data (XSLT), perhaps embedded in interactive browser web pages. However, this requires a prior design, something not yet developed for the files generated here. Moreover, XLST is somewhat involved and not convenient for the analysis required here. Instead, we developed a GUI interface which enables lower-level access to the XML data and provides an easier development vehicle for the kinds of exploration needed here. Lessons learned from this interface can guide future development of applications using XML-tagged documents.

Our development environment (known as XMLPartner) provides powerful tools for low-level access to the XML data. A well-structured XML file has the form of a completely hierarchical tree, wherein nodes contain the data and the attributes.[2] In our system, an XML file of any size (with extremely large files using a buffered stream) is loaded with one statement. Similarly, a search for nodes providing the answer to some query (the XPath expression conforming to the XML Path Language) is accomplished with one statement. This enables us to focus on development of queries and examination of search results (perhaps with further search statements). We have developed surrounding GUI components to facilitate examination of different aspects of the XML data (referred to below as XML Analyzer).

5.1.1 Global Examination of Data

In the first place, we used XML Analyzer to examine (and sometimes extract) interesting phenomena in the text. XML Analyzer can be used as a concordancer; a suitable XPath expression can extract all sentences in our TREC XML files (80 MB) that begin with "After" in four minutes. Similarly, we can find all discourse entities that contain a capitalized word, to examine whether we have assigned them an appropriate named entity type. In general, we use this basic capability to examine words, the entities and sentences in which they occur, and their attributes.

We display results of a search with the entity (if requested), the document title (the document number for TREC documents), and the sentence containing the entity. When we are searching only for sentences, no entity is given. A user can select a sentence and ask to see all the entities in that sentence. A user can select an entity and request all other entities which co-refer to it or have it as an antecedent.

5.1.2 Detailed Investigation of Discourse Entities

The XML Analyzer can be used to examine details about particular discourse entities. For example, question 1502 asks "when was President Kennedy killed". In the NIST top 10 documents for this question, a search on "Kennedy" in discourse entities identified 152 occurrences (the Kennedy clan). Narrowing the search to those also containing "Edward" gave 7 instances; expanding this to include entities where "Edward" was contained in the antecedent attribute identified an additional 14 instances. An examination of the attributes of these 21 instances showed 14 as the subject, one as the object, one as a possessive pronoun and three as a genitive determiner. and two as a prepositional object.

Use of the XML Analyzer in this way suggests that a user can examine the different relations in which an entity participates. For those as subject, we can examine the verbs to determine what kind of actions the subject performs (for action verbs) or what properties the subject has (for stative verbs). For those as possessive pronoun or genitive determiner, the user can examine what kinds of possessive relationships the entity can have (e.g., as brother, his back, or his commitment). For those as prepositional object, we can examine the relations the entity has with other entities. More generally, this suggests the possibility of an interactive web page allowing a user to explore the different relations in which a discourse entity participates, perhaps moving to other discourse entities with which it shares a relation.

5.2 Step 2: Answering Questions with XPath Expressions

As our first step in developing techniques for answering questions, we examined whether the

[2] Indexing of XML documents includes traditional indexing for information retrieval, but is also "XML-aware", meaning that searches can be efficiently performed on any XML tags, attributes, and values. We envision that XML output generated by our system would be subjected to XML-aware indexing.

answers (as contained in the patterns) occurred as discourse entities in our XML output. For virtually all cases, the answers were present in distinct entities; in those where they were not, we identified several bugs we were able to correct in our XML output processing.

This process generalizes well with our interface: create an XPath expression, determine whether it leads to appropriate discourse entities, and if not, make changes in some part of our system, either correcting bugs or altering our XML representation. This process has involved learning the intricacies of XPath expressions, which have proved capable of returning the exact answer to almost all TREC questions.

We developed XPath expressions for a contiguous 20 percent sample of the TREC questions, providing a basis for drawing conclusions. In general, the XPath expressions are highly confirmatory of techniques developed over the years in the QA track. The XPath expressions show that simple string patterns are quite effective and that syntactic and semantic information can be quite useful. Our development of these expressions shows that characterizing the patterns in the underlying text via XML elements and attributes is worthwhile for QA, and potentially other applications. We demonstrate this by showing the XPath expressions for several question types. In each of these cases, the development of an XPath expression proceeds by (1) further characterization of the question type, (2) development of a query component that selects segments, and (3) refinement of the query in specifying characteristics of the discourse entity.

5.2.1 WhatIs and WhatNP Questions

What questions have the highest frequency, constituting more than 40 percent of the questions, and have the most subtypes. Four principal varieties are: (1) "**What (is|was) (the NP ... | NP called | the ORD NP | NP1's NP2 | NP)?**", where NP is a noun phrase and **ORD** is an ordinal (e.g., 'first'); (2) "**What NPA (is (NP2 | PP) | did (NP2)? V (PREP)?)**", where **NPA** is **NP1** or **NP1's NP3**, **PP** is a prepositional phrase, **PREP** is a preposition, and the internal '?' indicates an optional element; (3) "**What is NP's (real | original | nick) name?**", and (4) "**What (do NP V | does NP stand for)?**", where V is a verb.

For the most general variety ("**What (is|was) the NP ... ?**"), a canonical answer would be "**X (is|was) the NP ... ?**". Examples are "What is the oldest college bowl game?" (1529), "What is the most populated country in the world?" (1544). and "What is the text of an opera called?" (1583). A suitable XPath expression can ask **//segment[contains(.,'(is|was) the NP ...')]**, i.e., a simple string match, or perhaps suitable subsets of the NP. To get at the specific discourse entity, the XPath expression would continue with **//discent[contains(.,'NP') and @synrole='obj'] /preceding-sibling::verb[.='was'] /preceding-sibling::discent[@synrole='subj']**, which says "find a discourse entity containing NP with syntactic role 'object' that is preceded by a verb equal to 'was' and that is preceded by a discourse entity with syntactic role 'subject'". This discourse entity is the answer to the question.

Another possibility for the general "**What (is|was) the NP ... ?**", as well as the third variety above "**What is ... name?**" and the second alternative of the fourth variety "**What does NP stand for?**", is a search for a relative clause, appositive, or parenthetical. As mentioned earlier, our text analysis and XML-tagging modules generally identify these as subsegments. Our segment search for these can be formulated as **//segment[contains(.,'NP') and (child::segment or contains(.,', or') or contains(.,'('))]**, which looks for a segment that contains the NP and contains either a nested segment or a simple string (a comma and "or" or an opening parenthesis). In these cases, the desired discourse entity would be obtained by first looking for **//discent[contains(.,'NP')]** and then either **/preceding-sibling::discent** or **/following-sibling::discent**. In the case asking what something stands for, the NP is usually an abbreviation or acronym. In this case, it is possible to build a more elaborate XPath expression that tests whether thc letters of the answer node(s) correspond to thc NP. With our low-level access to the answer nodes, however, it may be more efficient to perform this test in a post-processing phase that evaluates the answers.

A post-processing phase becomes even more important in handling the second variety listed above ("**What NPA (is (NP2 | PP) | did (NP2)? V (PREP)?)**"). For this variety, the segment search is generally of the form **//segment[contains(.,'NP2') and (child::verb[@root='V'] or contains(.,'PP'))]**, looking for elements of NP2, the root form of V, or elements of the prepositional phrase, usually the prepositional object, depending on the specific subvariety. The desired discourse entity may be present in the same discourse entity containing NPA (**//discent[contains(.,'NPA')]**), such as q1525, "What university did Thomas Jefferson found?", where our text processing created a single named entity "University of Virginia". However, in the more general case, the head noun of NPA would be a hypernym of an answer (such as q1499, "Which African country's major export is coffee?"). In this case, we might return all discourse entities in segments containing "major", "export", and "coffee" (**//discent**, i.e., without any further restrictions) and in post-processing ask whether any of them are hyponyms of "African country", using WordNet as the basis for such tests.

5.2.2 When Questions

When questions comprise 20 percent of the TREC set. In our analyses, only two varieties were needed: (1) "**WHEN (was|did) NP VP?**", where **VP** includes a verb and other constituents, and (2) "**WHEN was NP V?**". **WHEN** can be either "When", "What year", or some other time question. Although the second variety is a subset of the first, the second is distinguished in having a verb as the final element (e.g., q1502, "What year was President Kennedy killed?").

Since our parser's grammar labels time expressions, identification of time patterns for our segment search was not necessary. Instead, for the first variety, the XPath expression was generally **//segment[contains(.,'NP') and contains(.,'VP')]**. For the second variety, synset expansion of the verb was necessary, **//segment[contains(.,'NP') and (contains(.,'V') or contains(.,'V1') or ... contains(.,'VN'))]**, where the **VI** are verbs from V's synset. To get at the discourse entity, **//discent[@tag='when']** was sufficient.

5.2.3 Where Questions

Where questions constituted 10 percent of the TREC set. Most of these questions followed the pattern "**Where is NP**". There were some minor variations, e.g., with "where" replaced by "**At what place**" or with "**What NP ... LPrep?**", where **LPrep** is a locative preposition. In all cases, the segment search was specified by **//segment[contains(.,'NP')]** and the discourse entity search by **//discent[@tag='where']**.

5.2.4 Other Question Types

Who questions comprised more than 10 percent of the TREC set. Their complexity was comparable to the **What** questions: the discussion above on those questions generally covers the issues involved in the **Who** questions. **How** questions, including "how many", "how much", and "how measured", comprise the remainder of the TREC set. The XPath expressions for these questions generally follow the patterns for the **When** and **Where** questions, replacing the **tag** value to "howmany", "num", "howmuch", or "howmeas", which were created during the text analysis.

6 TREC-11 Results and Analysis

6.1 Official Results Using Document Databases

CL Research submitted 2 runs for the main task, both using the document database approach used in previous years. Our intent was that one run would be based on the top 10 NIST documents and the other based on the top 20. However, an error in submission resulted in the set for the top 20 being submitted twice. Our official confidence-weighted score (CWS) was 0.049, with 36 correct answers, 10 inexact answers, and 2 unsupported answers.

Our official submission was significantly affected by an oversight in which *Associated Press Newswire* texts were not properly subjected to the sentence splitter. The effect (for 101 questions) was that whole paragraphs were evaluated and scored as a single sentence. The scoring used in our system gave a large number of these paragraphs unduly high scores. These paragraphs, from which an answer was extracted, were thus given a high ranking, significantly affecting the CWS. We have not yet reprocessed those texts to determine the overall effect on our document database submission. By changing the scores for these answers to the average of our scores for *Xinhua* and *New York Times* documents, the estimated CWS was changed to 0.080. However, the effect is likely to be more significant, since the selection of these paragraphs as answer sources precludes the possible selection of correct answers from lower ranked passages.

Notwithstanding this difficulty, our document-based question-answering produced results consistent with our system's performance in the past two years. We calculated the mean reciprocal rank for our exact answers (0.128) and for the sentences (0.232) containing them. As indicated earlier, we made no significant changes in our document-based question-answering, so these results were expected.

6.2 Unofficial Results Using XML-Tagged Documents

To assess the potential benefit from using XML-tagged documents, we selected a contiguous set of 100 questions (1493-1592) and developed XPath expressions by hand for them to determine if we could obtain exact answers. This set was started after we had gained some familiarity with using the tagged documents and our XML Analyzer. We have not yet automated the creation of XPath expressions; we have found it necessary to develop an understanding of the patterns suitable for the different question types and varieties, as described in the previous section. We selected a contiguous set to compare our results to a contiguous subset of the full set of questions, rather than a random subset that might not generalize.

We applied the XPath expressions against the XML-tagged files for these 100 questions, first against the top 10 documents and then against the next 10 documents if we did not obtain an answer against the first 10. We constructed an answer set file conforming to the NIST specifications, using the first answer returned or NIL if no answers were returned. We did not use any scoring system to order the answers, but rather gave a score of 1005 to all non-NIL answers and 1000 to all NIL answers. As a result, sorting the answers by score ordered the answer file with the highest question number first. This answer file was then scored with the NIST Perl script.

Since many questions had no answers in the top 20 documents, we also formed a subset of 75 questions for which answers were present, but including the six questions in this subset for which no answers were present in the TREC collection, to test the effect of posing an XPath expression to the top 20 documents.

We developed the XPath expressions for these questions to conform as much as possible to linguistic intuitions, rather than just attempting to get the correct answer so that we will be able to develop appropriate mechanisms for automating the process. For the most part, the expressions have the simplicity described in section 5, with only a few requiring complex expressions. The expressions had a very high specificity in retrieving answers. The 75 XPath expressions returned a total of only 171 answers (2.3 per question), of which 97 were exact answers (1.3 per question). (For q1587, "What did Sherlock Holmes call the street gang that helped him crack cases?", which NIST characterized as not having an answer, the XPath expression returned "Baker Street Irregulars", although this answer would have been judged as unsupported.) Table 1 shows the confidence-weighted scores based on our official submission and based on the XML-based answers.

Table 1. Confidence-Weighted Scores for Question Samples

Sample	CWS
Official (100)	0.192
XML-based (100)	0.816
Official (75)	0.266
XML-based (75)	0.869

As can be seen, the question subset we have chosen is much better than our official results for the full set (0.049). In table 2, we show the mean reciprocal rank for these subsets.

Table 2. Mean Reciprocal Ranks for 75 Question Sample

Sample	CWS
Official (first answer only)	0.160
XML-based (first answer only)	0.800
Uofficial (top 5 answers)	0.243
XML-based (top 5 answers)	0.828

In this table, the first two rows correspond to the percent of answers that are correct, while the second two rows consider the top 5 answers, as in previous years. These results, again, are higher than our overall results, where we answered only 36 questions correctly and our overall mean reciprocal rank was 0.128. Thus, this sample may overstate how much we would achieve with XPath expressions for all 500 questions.

In general, our results using the XML-tagged documents and XPath expressions were quite surprising. While our XML-tagging is comprehensive, it is far from complete, as indicated above. In addition, the question types and varieties did not seem to require an elaborate typing of answers (such as Harabigiu et al. (2002) or Hovy et al. (2002)). Rather, the XML-based approach seems closer to the pattern-matching methods described in Soubbotin (2002), Brill et al. (2002), and Ravichandran & Hovy (2002), with additional benefits achievable by having structural information available. However, it is not clear how much taggging is necessary for question-answering. This is an issue for further research; we believe our methodological approach is well-suited to examining this issue, using the many levels of detail available.

7 Future Developments

As mentioned earlier, many components of our XML-tagging system can be improved, including our discourse analysis, anaphora resolution, semantic typing, and disambiguation components. As these improvements are made, they can be examined specifically for their contribution to question-answering. In addition, we see the XML-tagging approach as having potential benefits for investigation of linguistic phenomena, information extraction, novelty detection, and text summarization.

We will be generalizing our XML Analyzer to handle arbitrary tagging systems used in tagging text, such as part-of-speech taggers, chunkers, word-sense taggers, and discourse taggers. This will entail only minor changes and will facilitate examination of linguistic phenomena, including the possibility of adding tags, one of the basic objectives of XSLT.

In developing XPath expressions to answer questions, the final component of the expression requests discourse entity nodes with specific properties. By focusing instead on all discourse entities having particular properties over a range of documents, the XML Analyzer can be reconfigured to act as an information extraction tool. A specification of the discourse entity type desired to propagate a database can be used to build suitable XPath expressions to extract this data.

The TREC top documents were noteworthy for frequently containing the same or a very similar document several times (perhaps differing only in the document number). The low-level functionality available to us for examining XML nodes makes it easy to recognize such duplication. This can be extended to recognize near duplication based on varying criteria, such as synonymy. Using the Kennedy example described earlier, for example, it would be straightforward to examine the various relations in which Kennedy participates for synonymy and novelty.

Finally, the low-level functionality also allows us to summarize the characteristics of a text at any node level (e.g., frequency, types of nodes, and novelty). These characteristics can then be used to create various text summaries, and indeed, to create new XML documents by combining nodes from the original document. For example, encyclopedia articles frequently discuss a topic by defining it in several places using a copulative and possessive properties; the corresponding nodes from the original article can be used to generate an overall definition.

8 Summary

CL Research has made a preliminary investigation of the feasibility of massive XML tagging of source documents for the purpose of answering questions. Our results strongly suggest that this is a viable approach. Further, the development of the infrastructure necessary to evaluate this approach suggests that XML tagging may be useful in several other text processing tasks.

References

Brill, E., Lin, J., Banko, M., Dumais, S., & Ng, A. (2002). Data-Intensive Question Answering. In E. M. Voorhees & D. K. Harman (eds.), *The Tenth Text Retrieval Conference (TREC 2001)*. NIST Special Publication 500-250. Gaithersburg, MD., 122-131.

Harabigiu, S., Moldovan, D., Pasca, M., Surdeanu, M., Mihalcea, R., Girju, R., Rus, V., Lacatusu, F., Morarescu, P., & Bunescu, R. (2002). Answering complex, list, and context questions with LCC's Question-Answering Server. In TREC-10 Question-Answering. In E. M. Voorhees & D. K. Harman (eds.), *The Tenth Text Retrieval Conference (TREC 2001)*. NIST Special Publication 500-250. Gaithersburg, MD., 355-361.

Hovy, E., U. Hermjakob, & C. Lin. (2002a). The Use of External Knowledge in Factoid QA. In E. M. Voorhees & D. K. Harman (eds.), *The Tenth Text Retrieval Conference (TREC 2001)*. NIST Special Publication 500-250. Gaithersburg, MD., 644-652.

Linguistic Data Consortium (2002). The Rhetorical Structure Theory Discourse Treebank. ISBN 21-58563-223-6. Philadelphia, PA.

Litkowski. K. C. (2001). Syntactic Clues and Lexical Resources in Question-Answering. In E. M. Voorhees & D. K. Harman (eds.), *The Ninth Text Retrieval Conference (TREC-9)*. NIST Special Publication 500-249. Gaithersburg, MD., 157-166.

Litkowski, K. C. (2002a). CL Research Experiments in TREC-10 Question-Answering. In E. M. Voorhees & D. K. Harman (eds.), *The Tenth Text Retrieval Conference (TREC 2001)*. NIST Special Publication 500-250. Gaithersburg, MD., 122-131.

Litkowski, K. C. (2002b). Digraph Analysis of Dictionary Preposition Definitions. *Proceedings of the ACL SIGLEX Workshop: Word Sense Disambiguation*. Philadelphia, PA., 9-16.

Ravichandran, D. & E. Hovy. (2002). Learning Surface Text Patterns for a Question Answering System. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, PA., 41-7.

Soubbotin, M. M. (2002). Patterns of Potential Answer Expressions as Clues to the Right Answer. In E. M. Voorhees & D. K. Harman (eds.), *The Tenth Text Retrieval Conference (TREC 2001)*. NIST Special Publication 500-250. Gaithersburg, MD., 122-131.

CLARIT Experiments in Batch Filtering:
Term Selection and Threshold Optimization in IR and SVM Filters

David A. Evans, James Shanahan, Norbert Roma, Jeffrey Bennett, Victor Sheftel,
Emilia Stoica, Jesse Montgomery, David A. Hull, Waibhav Tembe

Clairvoyance Corporation, Pittsburgh, PA

1. Introduction

The Clairvoyance team participated in the Filtering Track, submitting two runs in the Batch Filtering category. While we have been exploring the question of both topic modeling and ensemble filter construction (as in our previous TREC filtering experiments [5]), we had one distinct objective this year, to explore the viability of monolithic filters in classification-like tasks. This is appropriate to our work, in part, because monolithic filters are a crucial starting point for ensemble filtering, and it is possible for them to contribute substantially in the ensemble approach. Our primary goal in experiments this year, thus, was to explore two issues in monolithic filter construction: (1) term count selection and (2) filter threshold optimization.

In fact, our pre-TREC experiments were conducted in a brief period and we were unable to complete all the tests we had planned. Our official submissions reflect essentially our first, baseline results. They are overall poor in comparison to other results reported this year.

However, an additional focus of our work relates to the general problem of exploiting training data, in particular, where there are only a few positive-example documents for a topic. We regard such cases as more realistic (e.g., in commercial settings) than the categorization-oriented tasks we have seen in TREC filtering in the past, e.g., based on the Reuters collection. Thus, in a series of follow-up experiments, we explored the strengths and limitations of classifier-based approaches (using kernel methods) and CLARIT-IR-based ones on the fifty TREC-2002 "Assessor" Topics.

In our CLARIT-IR-based experiments, we aimed to establish a more accurate baseline than the one reflected in our official submissions. We also sought to vary the term-extraction techniques we used, to optimize performance on a topic-by-topic basis.

In our kernel-based (SVM) experiments, we used non-mathematical (non-QP) based approaches to learning SVMs. We used both NLP-based features and simple white-space-delimited ones; and we developed a preliminary approach to thresholding the classifier margin.

In the following sections we first describe our official submitted runs and results and then present in greater detail the post-TREC experiments that we conducted.

2. Official Batch Filtering Runs

Our official batch filtering runs reflected a straight-forward extraction of term vectors from positive training documents, the setting of thresholds based on calibration of the term vectors over the training data, and the use of the term vectors to score (retrieve/rank) documents in the test collection.

2.1. Preparing Filters and Testing

As a general approach to handling the available training data, we divided the training corpus equally into two parts, one half of which we used for constructing filters (i.e., extracting terms, assigning weights, and determining the optimal term profile cutoff), the other half of which we used for validation (including score-threshold setting). We constructed monolithic filters for each topic automatically, based on the positive examples of each topic. In fact, we used a slightly modified version of the training database: we added two additional, identical positive example "documents" for each topic. These were created by the system from the topic's title and its short and long descriptions, with all meta-language ("Retrieve documents which...," "Find documents that...") automatically removed. We chose to add these artificial documents to increase the number of training documents and to emphasize terms that we anticipated would be especially useful in the filter.

Terms for all topics in this combination of positive examples and artificial documents were generated by CLARIT NLP (yielding morphologically normalized single words, phrases, and sub-phrases), then weighted, ranked, and selected for extraction (to represent the term profile in the filter) using our thesaurus extraction method "Prob2" (as given in Figure 1). We used Prob2 as our default and only term-extraction method based, in part, on our observations of Prob2's overall robust performance compared to other term-extraction methods in our TREC 2001 experiments. While keeping the method of selecting terms for topics constant, we experimented with optimizing the number of terms for a given topic.

We investigated several techniques for determining how many terms to include in the filter for a given topic. We settled on a method based on the 2nd derivative of a topic's term weight profile (*w''*). In short, this approach examines a set of terms ranked according to term weight, and disregards all terms occurring after the point where the term weight profile begins to level off. This point is determined by the condition $0 > w'' > \varepsilon$. (In our case, we set ε to 0.01.) In this way, all terms that did not show evidence of being particularly characteristic of a topic (according to their rank in the term weight profile) were disregarded. We applied this method of term count selection to construct filters for each topic. We imposed the additional condition that no topic filter use fewer than five terms. For each of our submitted runs, the average number of terms in a filter was 26, and the maximum was 104.

Once we established which terms (and how many) to use in constructing a filter, it remained for us to determine the threshold for each topic. This was one of the chief issues we wanted to explore, and so we took a different approach for each of our submissions. In our runs CCT11BFC and CCT11BFD, the filter was applied to the entire training corpus. That is, the threshold was optimized over both the first half of the corpus, which we used for constructing the filter, as well as the second half, which had thus far been unused. For CCT11BFC, the filter's threshold was set using the beta-gamma method on normalized linear utility, T11SU, to decrease the likelihood that we would over-fit the training data. (Cf. [13;14] for discussion of the beta-gamma threshold setting method.) For this run, beta-gamma values were 0.1 and 0.4, respectively. To further decrease the possibility of over-fitting the training data, we employed an additional "global threshold multiplier" that relaxed the optimal threshold a bit further. For run CCT11BFC, this global multiplier was set to 0.95.

Our other submission, CCT11BFD, was identical to CCT11BFC, with two exceptions. For this run we employed no beta-gamma regulation at all, and instead lowered the global threshold multiplier to 0.85. Finally, all filters for both CCT11BFC and CCT11BFD were run on the full testing set.

2.2. Official Test Results

Table 1 presents a summary of various batch filtering runs in terms of normalized linear utility (T11SU) and F-Beta. Row one of this table gives the median of all submitted runs from all groups for TREC-2002 batch filtering. The second and third rows summarize the results for our two submitted runs. The remaining rows show results for other unofficial runs we completed, including an Adatron SVM [1;6;11] run.

$$Prob2(t) = log(R_t + 1)\ x \left(log(\frac{N - R + 2}{N_t - R_t + 1} - 1) - log(\frac{R + 1}{R_t} - 1) \right)$$

$$Rocchio(t) = IDF(t)\ x\ \frac{\sum_{D: DocSet} NTF_D(t)}{R}$$

$$RocchioFQ(t) = IDF(t)\ x\ \frac{\sum_{D \in DocSet} TF_D(t)}{R}$$

$$GL2(t) = \frac{4xR_t xN_t}{(R + N_t)^2}$$

N is the number of documents in the (reference) corpus; N_t is the number of documents in the (reference) corpus that contain term t; R is the number of documents (for training or feedback) that are relevant to the topic; R_t is the number of documents (for training or feedback) that are relevant to the topic and contain term t; TF is the (raw) frequency of term t in a document; and NTF is the normalized frequency of term t in a document.

Figure 1. Term-extraction formulae

Run Description		T11SU	F-Beta
Median for all Submitted Runs		0.316	0.129
Submitted Results	CCT11BFC	0.186	0.147
	CCT11BFD	0.184	0.145
Unofficial Results	CCT11BFA	0.147	0.130
	CCT11BFB	0.165	0.129
	Adatron	0.328	0.035

Table 1. Results of official and pre-TREC batch experiments (on all 100 topics)

2.3. Observations on Official Runs

On the whole, our official results were unsatisfactory. Our failure to perform well may have been due to several factors. We may have selected terms poorly for various topics, and therefore had a poor characterization of these topics. It was also possible that we chose appropriate terms, but too many or too few of them. Finally, we may have correctly chosen our terms and term counts, and still have performed poorly on various topics due to poor thresholding. We did several analyses to see which of these factors actually was responsible for our weak performance.

As we investigated topics where we performed poorly, we saw little indication that we had grossly erred in our method of choosing which terms to extract from positive examples. Thus, our decision to use a single feature-extraction method (Prob2) that had performed well and robustly in the past does not seem to have harmed our effort significantly.

Additionally, there was little evidence that our term count optimization was faulty. We conducted post-submission experiments where we added terms to (poorly performing) topic profiles in which we had originally used relatively few terms. We likewise did experiments where we removed terms from topic profiles in which we had originally used many terms. In neither case did we see a dramatic change in the performance of filters upon the addition or removal of terms from the profile.

We did see, however, that setting filter thresholds improperly had a remarkable impact upon a filter's performance. In particular, we observed that for many of the topics where we performed poorly, we had set the filter threshold much too low, thereby allowing for the retrieval of many non-relevant documents. Results of our post-submission experiments indicate that the negative effects of poor thresholding far outweigh the positive effects of good term and term count selection.

We saw this principle at work, for example, in Topic 144, *Mountain Climbing Deaths*, which was one of the topics on which we performed extremely poorly. Upon examining the actual terms (and number of terms) in the profile, we could see that they were a fair characterization of the topic. Our recall figure was quite high (0.965—we retrieved 55 of 57 total relevant), and initial precision was quite high: roughly the first third of the documents we retrieved were relevant—a good indication that our terms and term weights were on target. The explanation for such poor performance, then, can be found in our set precision figure (0.026): we retrieved far too many non-relevant documents (2048 out of 2104).

Furthermore, we observed the positive effect that conservative thresholding can have in overcoming the lesser negative effects of poorly chosen terms or term counts. We saw this in some topics where we performed quite well in comparison to the TREC median (T11SU), even though many of the highly ranked documents were not relevant. Our good performance (relative to the median) is likely the result of choosing terms that were at least adequate, and, especially, having a threshold that was conservative enough to prevent over-delivery of non-relevant documents. Topic 122, *Symptoms Parkinson's Disease*, was one topic where we observed this behavior.

The results of our analyses, then, clearly demonstrate that having good terms and term counts is outweighed by setting an improper threshold. On the other hand, accurately choosing a proper threshold helps even in instances where term and term count selection are not especially good. The greater danger lies in using a threshold that is too relaxed rather than setting too conservative a threshold. Thus, our decision to override and *lower* the threshold for each topic set automatically on the training data was the principal cause of our poor overall performance.

3. Post-TREC Experiments: IR-Based Filters

We were naturally interested in assessing the problem of threshold setting in our post-TREC follow-up experiments. In particular, we wanted to establish our baseline performance in threshold setting and to look more closely at the problem of term selection.

We confined our evaluation to the first fifty ("Assessor") topics, because they proved to be the most valuable (and valid) ones in the test suite, and because these topics also seem more realistic than the artificially generated "Intersection" topics. In our subsequent analysis, we report both our post-TREC results and official TREC results on Assessor topics only.

3.1. Revised (Corrected) Term/Threshold Selection

In our first post-TREC experiment, we repeated our basic TREC runs with "normal" threshold setting. That is, we did not force the threshold (set on the training data) to be more relaxed when running on the test collection. This experiment used only the simplest approach to term selection (based on Prob2 extraction and 2nd-derivative term-count selection), threshold calibration on the full training database (using beta-gamma threshold setting), and direct ranking of the test collection. We called this run *Prob2-2D*.

In our second post-TREC experiment, we first split the training data into halves and used one half (including approximately half the positive examples) for candidate term selection and the other half for validation. In this approach, we were interested in trying several different term-extraction methods and predicting which method would give the best terms for each topic. Thus, we used each method (and 2nd-derivative term-count selection) on the positive training documents for a topic in the first half of the training corpus to create a term vector for each topic, and then tested the performance of each vector against the second half of the training corpus. Based on which vector gave the best performance (T11SU score), we chose the term-extraction method used to create that vector as the "best" for that topic. We then repeated the procedure in our first post-TREC experiment, but with the term-extraction method set to the "best" method for each topic. We called this run *Opt-2D*.

The steps in our process are given in Figure 2. Note that the split of the training data into halves (or any other arbitrary proportion) to yield a sub-corpus for topic modeling (term extraction) and a sub-corpus for validation (testing a model), is based on a pseudo-random assignment of documents to one or the other portion. This means that, in the case of some topics, there might be very few positive examples of a topic in any one of the training sub-corpora. A paucity of data can lead to poor training, of course, but we decided not to intervene to insure optimal splits in training data precisely because we wanted to assess, as well, the robustness of our generalized topic-modeling process.

1. Split the training set. First, sort (scramble) the document ids. (For a database with 10 docs, such "scrambling" might produce: 0 2 4 6 8 1 3 5 7 9.) Next, apply the desired Training/Validation split. (For the TREC experiments, the split is 50/50, based on choosing every other document for a split.) Pick one split for Training, one for Validation. (In the TREC experiments, the 2nd half ("odd" documents) was chosen for Training/term-extraction and the 1st half ("even" documents) was chosen for validation/optimization.) The pre-scrambling makes it possible to select any subset with reduced bias. (If one chose a 60/20/20 split for the 10-document collection, above, the system would deliver the subsets "0 3 6 9 1.4", "7 2," and "5 8".)

2. Choose extraction method. If the extraction method is fixed (e.g., Prob2), skip this step and go to Step 3. For each candidate extraction method (e.g., Prob2, Rocchio, RocchioFQ, and GL2), create a filter using the Training half of the training corpus. Optimize the term count by applying the 2nd derivative method. Choose Max(MinimumTermCount, 2ndDerivTermCount). (The MinimumTermCount used in post-TREC experiments is 10.) Truncate the term vector to the specified term count. Set the threshold using beta-gamma optimization ($\beta=0.1,\gamma=0.4$) over the entire training set. Retrieve over the Validation half of the training set (using the optimized threshold) and compute utility (T11SU). Choose the extraction method with the highest score.

3. Extract final filter. Extract terms from the entire training set using the chosen method. Optimize the term count (using 2nd derivative, as described above), subject to the MinimumTermCount. Truncate the vector to that count. Set the threshold on the entire training set using beta-gamma optimization.

Figure 2. Procedure for creating a CLARIT filter profile

The formulae for our term-extraction methods—Prob2, Rocchio, RocchioFQ, and GlobalLocal2 (GL2)—are given in Figure 1. In both experiments, we used the full training corpus as the reference corpus. Processing time for these filters averaged 17 seconds per topic for training and testing *combined*. Filter length for Prob2-2D averaged 33.34 terms and for Opt-2D 15.76.

3.2. Post-TREC Experiment Results

As can be seen from the results in Table 2, both Prob2-2D and Opt-2D clearly out-perform our submitted (official) runs. (The values in Table 2 for our official runs reflect our performance on the Assessor topics only.) Compared to the median reported for the group on Assessor topics, both Prob2-2D and Opt-2D have lower T11SU scores. However, in terms of F-Beta, both post-TREC runs show rather impressive performance.

In our Opt-2D runs, the Prob2 extraction method was chosen only 6 times, whereas Rocchio was chosen 30 times, RocchioFQ 8 times, and GL2 6 times. In terms of individual-topic results, Opt-2D gave significantly better performance (>0.10 absolute difference in score) than Prob2-2D on 10 topics for T11SU and 9 topics for F-Beta. In contrast, Opt-2D was significantly worse on 11 topics for T11SU and on 7 for F-Beta. In the aggregate, however, the effect of term-extraction method optimization appears to be negligible.

Run Description		T11SU	F-Beta
Median for all Submitted Runs		0.377	0.234
Submitted Results	CCT11BFC	0.243	0.259
	CCT11BFD	0.243	0.259
Post-TREC Experiments	Prob2-2D	0.309	0.323
	Opt-2D	**0.315**	**0.326**

Table 2. Results of post-TREC-batch experiments

We note, however, that our choice of an optimum method was based on the performance of a candidate filter on half the training corpus. In those cases where we had poor training splits, our choice was not well informed. Clearly, this is an area for further work.

The overall strong performance on F-Beta for both runs confirms our hypothesis that the basic method we have used is robust and practical. It also confirms that the poor results in our official runs were due to improper threshold setting, in particular, our decision to relax the threshold values that were determined for filters on the training corpus.

4. Post-TREC Experiments: Kernel-Based Filters

In addition to our IR-based runs, we decided to expand our evaluation of kernel techniques for batch filtering in a series of post-TREC experiments. The essential questions we focused on include how well kernel methods perform on topics with limited training data and how flexible the learned thresholds can be when data is sparse. We explored both our kernel-Adatron and a new version of an SMO algorithm.

4.1. General Note on Kernel (SVM) Methods

Support vector machines (SVM) are a general purpose machine learning approach [2;12], with our interest being principally in learning classification models from labeled data. Our batch filtering SVM study was limited to learning a binary SVM classifier for each topic (positive and negative class). This corresponds to searching for (or learning) a hyperplane that provides maximum separation between the positive and negative training examples. Since text classification problems are of high dimensionality (which are generally linearly separable), it is sufficient to search for this hyperplane in the term/word space, thus avoiding the use of more complex feature spaces that can be induced easily using kernel (non-linear similarity) functions. Hyperplane selection is based upon ideas from statistical learning theory [12], where the hyperplane that is furthest away (has maximum margin) from all training data and that provides (tolerable) class separation is chosen. Large margin separation has been theoretically shown to lead to improved generalization.

More formally, SVM models or classifiers denote a separating hyperplane between two classes, whereby datapoints falling on one side of the hyperplane denote one class and datapoints falling on the other denote the other class. In linear kernel-based SVMs, hyperplanes are typically represented in *primal form* as follows (where <.,.> denotes inner/dot product):

$$Class\ (X) = Sign(\langle W, X\rangle + b)$$

where W is a weight vector, and b is the bias or threshold. See Figure 3 for a graphic depiction of a hyperplane for a linearly separable dataset. An alternative and more general representation of a hyperplane that is commonly used in SVMs is the following *dual representation*:

$$Class(X) = Sign\left(\sum_{i=1}^{L} \alpha_i y_i \langle X_i, X\rangle + b\right)$$

Here, the alphas (α_i) denote the Lagrange multiplier associated with each example. This representation permits the learning of such classifiers using well-known optimization techniques such as quadratic programming. After learning, only a small percentage of the training data will have non-zero Lagrange multipliers. These examples are known as the *support vectors.* For dot-product (linear) kernels the dual representation of a hyperplane can be mapped to the primal form, thus, yielding a computationally more efficient model, akin to the more traditional information retrieval model. The above dual representation of a hyperplane can be further generalized by considering different forms of the similarity function or kernels such as polynomial, LSI Kernels [4], and String Kernels [9]. For our current purposes of text classification, linear kernels were deemed to be sufficient.

4.2. Learning SVMs

Support vector machines are commonly trained using either mathematical programming (MP) approaches such as quadratic programming or by strategies that avoid the use of the MP techniques. The latter techniques have the added attraction of being easier to implement, while providing similar levels of performance as their MP counterparts. For our experiments, we implemented and evaluated two non-MP based approaches: the kernel-Adatron (KA) algorithm [1;6;11] and variations of the sequential minimal optimization (SMO) algorithm [10;7]. Both of the algorithms are outlined briefly below.

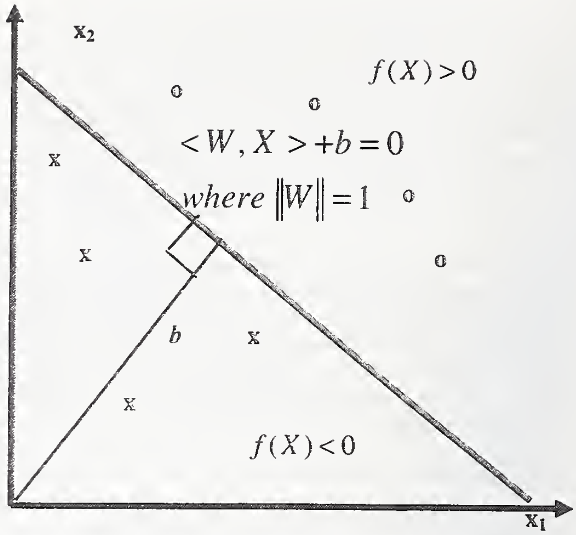


Figure 3. A linearly separable dataset in a two-dimensional space for a two-class problem (where the "o"s correspond to one class, and the "x"s denote the other)

Kernel-Adatron Learning Algorithm. One of the simplest strategies for learning a support vector machine is to update the Lagrange multipliers, α, associated with each example iteratively. This approach has been taken in the kernel-Adatron algorithm proposed by various researchers (cf. [6] and [11]). The Adatron was originally proposed by Anlauf and Biehl [1] in the field of statistical mechanics. It is an on-line learning algorithm for learning perceptrons. In [1], it was proved that the Adatron converges to a maximum margin solution; that is, the discovered hyperplane is a fixed point of the adaptive algorithm for linearly separable data. In [6] and [11], the Adatron algorithm was extended to learn the dual representation of a separating hyperplane in which the dot product is replaced with the more general kernel, thereby expanding the domain of application of the Adatron to non-linear problems. (A simplified version of the pseudo-code for the kernel-Adatron algorithm is presented in Figure 4.) We limited our implementation to training hard-margin SVMs.

SMO Learning Algorithm. The Sequential Minimal Optimization (or SMO) algorithm is an alternative method for training SVMs [10]. Traditionally, training an SVM required the solution of a very large quadratic programming (QP) optimization problem. SMO breaks this large QP problem into a series of the smallest possible QP problems, where only two Lagrange multipliers, α_i, are optimized at each iteration. Since only two parameters are considered at a time, while all others are fixed, it is possible to derive an analytical solution as opposed to the numerical methods used in MP solutions. This avoids using a time-consuming numerical QP optimization as an inner loop in the algorithm. On each iteration, SMO chooses two Lagrange multipliers to optimize jointly (typically the

```
1.  Given Training data S where each example $i$ is of the form $(x_{i,1},\ldots, x_{i,n},y_i)$, and a learning rate $\eta$
2.  Set $\alpha$ vector to zeros; (could use $b_o$=0)
3.  For i = 1 to |Train|
            $z_i = \sum_{i=1}^{|Train|} \alpha_i y_i \langle X_i, X \rangle$
4.  For i = 1 to |Train|
            1.  Let $\delta_i = \eta(1-y_i z_i)$ be the proposed change to the multiplier $\alpha_i$
            2.  If $((\alpha_i + \delta_i) \leq 0)$   set $\alpha i$ to 0  else $\alpha_i := \alpha_i + \delta_i$
5.  (if b used, b=0.5 (min($z_i$+) + max($z_i$-)) where $z_i$+ denotes those patterns $i$ with class label +1 and $z_i$-
    denotes those patterns $i$ with class label -1)
6.  If maximum number of presentations of the pattern set (epoch) has been exceeded OR (min($z_i$+) -
    max($z_i$-)) == 2.0
    then stop otherwise goto step 1
```

Figure 4. Partial pseudo-code for Kernel-Adatron algorithm

```
1.  Given Training data S where each example $i$ is of the form $(X_1,y_1,\ldots, X_n,y_n)$
2.  Set $\alpha$ vector to zeros; $b_o$=0 (the bias term);
3.  ExamineAll = true
4.  Compute error vector E; $E_i = -y_i$
5.  If ExamineAll then      //loop thru all examples
            For (int i =0; i < TrainDB.count; i++)
                    Set i2 to I; Use heuristics to find a partner i1
                    Try to optimize(alpha[i1], alpha[i2])
    Else   //loop thru all examples with non-bounded alphas
                For (int i =0; i < TrainDB.count; i++)
                        If alpha(i) > 0 and alpha(i) < C
                            Set i2 to I; Use heuristics to find a partner i1
                            Try to optimize(alpha[i1], alpha[i2])
    If (ExamineAll), then ExamineAll= false
    Else if (NumberOfUpdates == 0), then ExamineAll =1
6.  if more alpha updates are possible (i.e., ExamineAll or NumberOfUpdates > 0) Goto step 5
```

Figure 5. Partial pseudo-code for SMO algorithm

Decision Variable	Explored Values
Learning Algorithm	Adatron, SMO, SVM^{Light}
C (Upper bound for Lagrange multipliers)	3, 10
Learning Rate (Adatron)	0.75
Tolerance	0.001
Type of kernel	Linear
Class Ratio	1:4, 1:10 , use all training data
Sampling Strategy	Random
Term type	NLP; single words
Term Ranking Algorithm	• Use all terms • Mutual Information: Use k terms that have highest MI for each topic
Number of terms k	k = 1,000, 10,000, All
Term weighting	Normalized TF*IDF

Table 3. Decision variables and explored values for current experiments using SVM text classifiers

examples that have the largest polar error), finds the optimal values for these multipliers analytically, and updates the SVM to reflect the new optimal values. (A simplified version of the pseudo-code for the SMO algorithm is presented in Figure 5.) For our experiments, we implemented two variants of the SMO algorithms proposed by Keerthi et al. [7] that provide better heuristics for determining which pair of Lagrange multipliers to update next and that provide better stopping criteria. For our current study, two variations of the SMO algorithm were implemented and evaluated: SMOK1 and SMOK2, corresponding to modification 1 and modification 2, respectively, as proposed in [7].

4.3. Preprocessing

We examined two representations of documents: one using CLARIT NLP-based (single or multi-word) terms and the other using single white-space-delimited words. The latter approach involved the following steps: replace all numbers and punctuation by spaces; eliminate stopwords such as *articles* and *prepositions,* etc. In both preprocessing approaches each term is associated with a TF*IDF weight, where *TF* denotes the frequency of a term in a document, and *IDF* is calculated based on the distribution of the term in the training corpus. The TF*IDF weights were then normalized leading to documents vectors of unit length.

For some of our experiments we chose a subset of terms in the term-space of the training corpus. In such cases, we ranked terms based upon their mutual information with the class label and the *k* terms with highest mutual information were selected to represent each document. The mutual information *MI(x_i, c)* between a feature, x_i, and a category or topic, *c*, is defined as follows:

$$MI\ (x_i,c)=\sum_{x_i\in\{0,1\}}\sum_{c\in\{0,1\}} P\ (x_i,c)\ \log\frac{P\ (x_i,c)}{P\ (x_i)\ P\ (c)}$$

Following feature selection, the document vectors were again normalized to unit length.

A learning step follows where for each topic/class/ category a topic-specific binary classifier is learned from the training data that models the topic (positive class) and the *not*-topic (or negative class). While it is possible to learn a topic SVM classifier by using all available training data, it is computationally attractive to reduce the number of training data, especially the number of negative examples. We currently achieve this through random sampling of the negative class, though all explicitly labeled negative documents are used. Typically, given *n* positive training examples for a topic, we chose *m* * *n* negative documents. We explore different values of *m* in our experiments.

4.4. Experiment Results using SVMs

For each of our experiments, we trained a linear TF*IDF kernel-based SVM (i.e., linear kernel, where each term is weighted using TF*IDF) for each topic using the training data. For most machine learning processes, with SVM-based approaches being no exception, there are many parameters and decisions that need to be made in order to generate a model that performs well on unseen data. Some of these are domain specific (e.g., text vs. images), while others are algorithm specific (e.g., the upper bound for Lagrange multipliers, C). The domain-specific decision variables for text include the following: (1) the number of terms used to represent each topic; (2) the number of on-topic training documents; (3) the ratio of positive to negative documents; (4) the sampling strategy for the negative class; and (5) the representation of a document using single words or NLP-based terms. In an ideal setting one could potentially chose the optimal configuration for a topic using, for example, n-fold cross validation. However, due to time limitations, we were unable to carry out such experiments in our post-TREC work. Instead, we report results where the different experiment variables are set to equivalent values across all topics for a particular experiment. The decision variables and explored values for our experiments are presented in Table 3.

The results of the more interesting experiments are presented in Table 4, where each row denotes one experiment on 50 Assessor topics. We report the T11SU and F-Beta measures for each experiment. For our some of our experiments we used different document sampling strategies. One was based upon a sampling of positive to negative documents (denoted as a ratio in Table 3). The other was based on using all labeled documents and fixed sample size of all unlabeled documents in the training size (denoted as an integer in the *Class Ratio* column in Table 4). Experiments on the 50 Intersection topics were not carried out, apart from one experiment, which was performed with the kernel-Adatron algorithm using all NLP-based terms and a 15,000 sample of the training set, yielding a T11SU performance of 0.328 (and 0.342 on the fifty Assessor topics).

For each experiment, training a battery of 50 binary classifiers (one classifier corresponding to each Assessor topic) took approximately twenty minutes (or approximately 24 seconds per topic), while evaluation took approximately two to three hours. The CC SVM toolkit is developed in Java and experiments were carried out under Linux and Windows XP on a 866-MHz Pentium III computer with 1 gigabyte of RAM.

Among the results, the best overall performance was given by an SMOK2 run (*SMOK2-θ.45*) representing one of our first experiments in thresholding the margin scores given by the SVM. In particular, we found the margin that gave optimal T11U utility on a resample of the training corpus, $\text{Margin}_{\text{MaxU}}$, and used the following formula to compute the new threshold, $\theta_{\text{Opt},}$, where ThresholdDiscount was set to 0.45 for this run:

$$\theta_{Opt} = ThresholdDiscount * (Margin_{MaxU} + 1) - 1$$

Algorithm	C	Class Ratio	Term Type	# Terms	T11SU	F-Beta
SMOK1	3	4	SingleWds	1000	0.347	0.144
SMOK1	3	10	SingleWds	1000	0.356	0.129
SMOK1	3	4	SingleWds	10000	0.367	0.173
SMOK1	3	10	SingleWds	10000	0.368	0.170
SMOK2	10	14125	NLP	All	0.356	0.077
SMOK2	10	14125	SingleWds	All	0.373	0.142
Adatron	N/A	14125	NLP	All	0.341	0.053
Adatron	N/A	14125	SingleWds	All	0.366	0.149
SMOK2	10	7605	SingleWds	All	0.376	0.147
Adatron	N/A	7605	SingleWds	All	0.363	0.154
SMOK2-θ.45	10	7605	SingleWds	All	**0.408**	**0.271**

Table 4. Results of SVM experiments on 50 Assessor topics

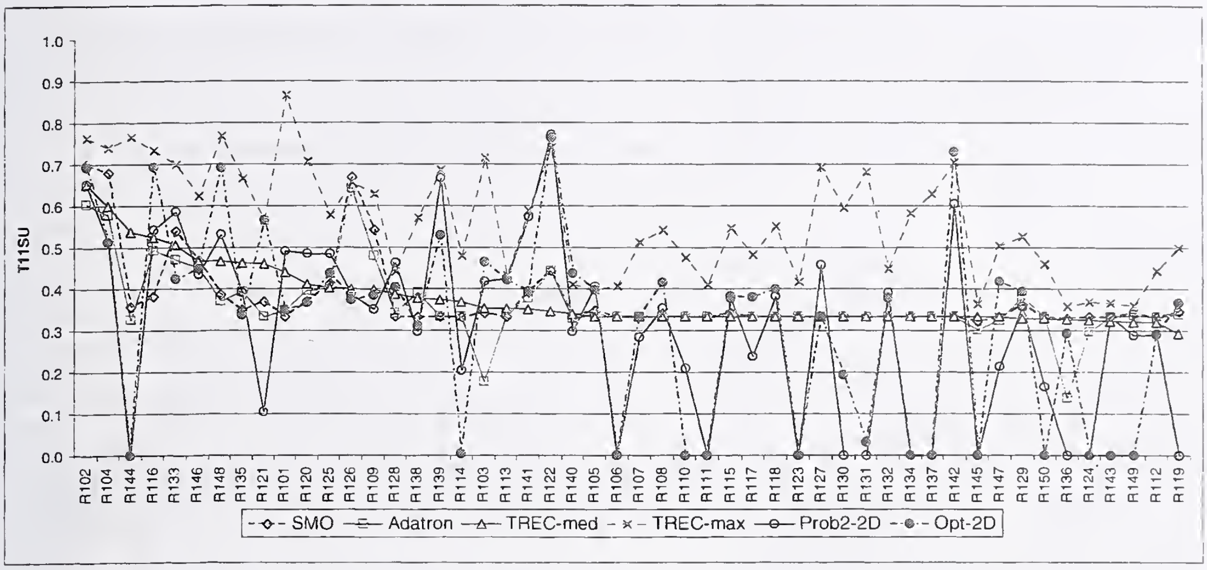


Figure 6. T11SU comparative results for SVM-and IR-filters on topics ranked by TREC median performance

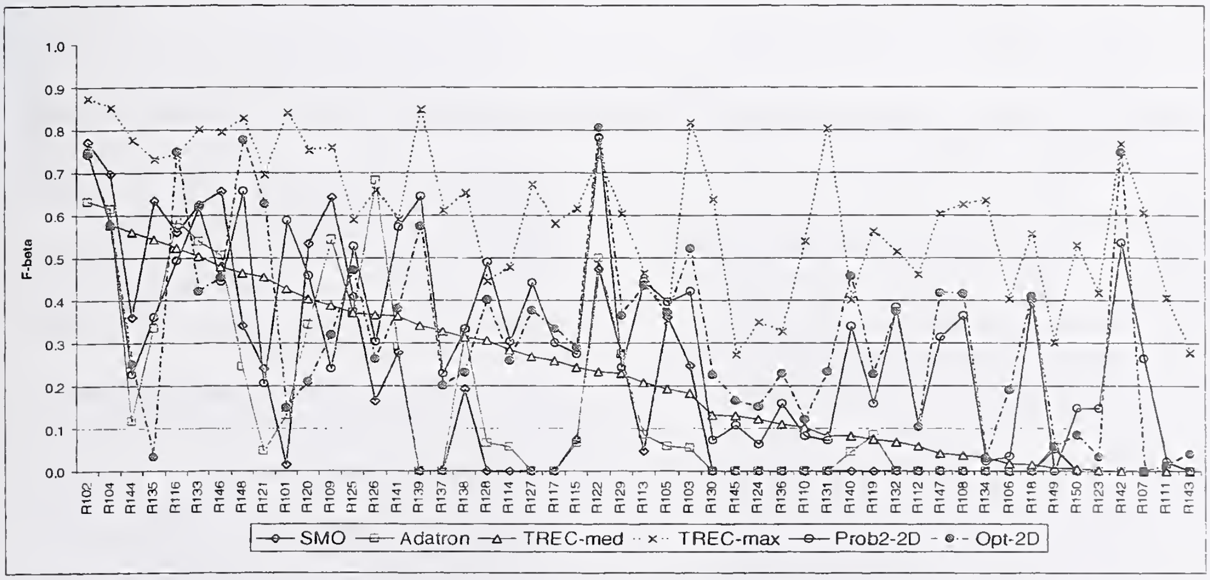


Figure 7. F-Beta comparative results for SVM-and IR-filters on topics ranked by TREC median performance

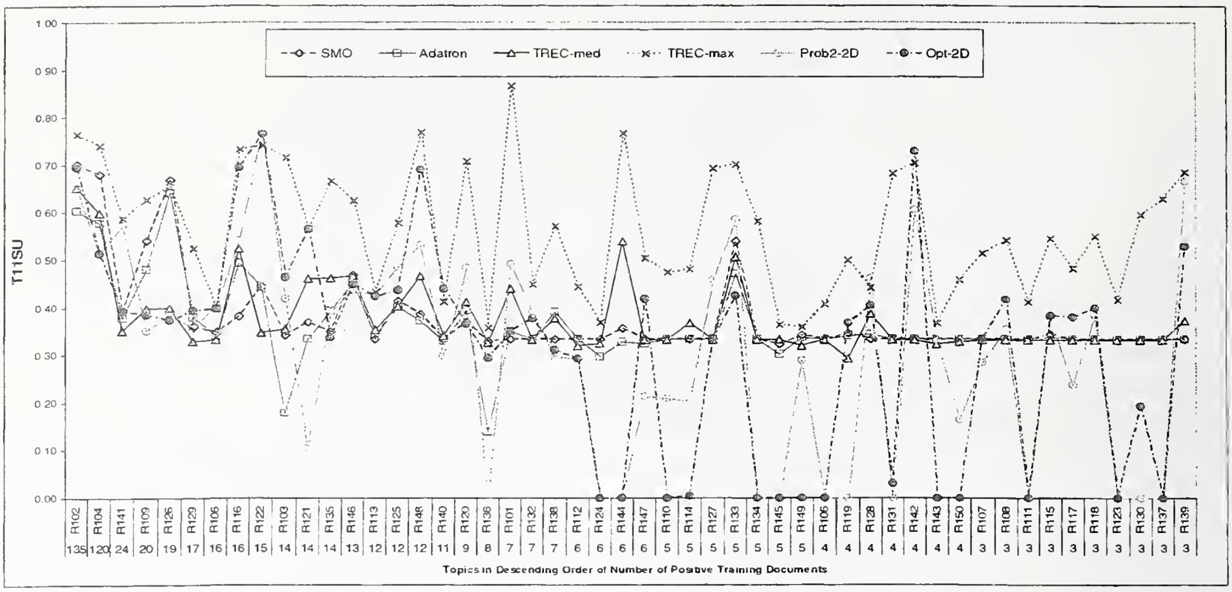


Figure 8. Comparison of SVM-based and IR-based filters: T11SU scores x Topic x Training Data

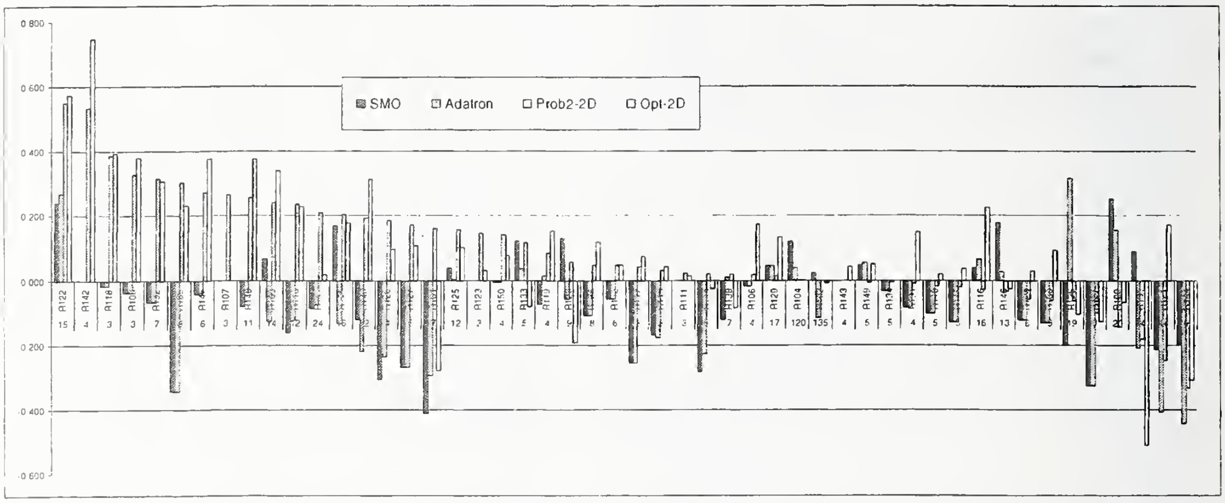


Figure 9. Difference from TREC med F-Beta-scores x Topic x Training Data, ranked by Prob2-2D–TREC med

4.5. Observations on SVM Filters

Overall, the performance of the learnt SVM classifiers is good compared to the submission results for other groups. The T11SU utility measure is average, while the F-Beta measure is low, apart from the SMOK2-θ.45 run with its more reasonable performance of 0.271. The lack of higher performance for the SVMs is partly due to the experimental setup, which did not employ cross-validation; i.e., for each experiment, each topic SVM was trained using the same parameter settings, thereby limiting potential performance of the learnt SVMs. Allowing the determination of a customized setting (potentially optimal) for each topic should lead to improved performance. In addition, our preliminary work on thresholding the margin value of the SVM output has given very encouraging results. This is consistent with results form other groups for this particular dataset [3]. However, for some other datasets in the past, this did not improve performance [8].

Given the results of our limited experiments, we can make the following observations:

- Using a simple tokenizing-based representation of a document (in lieu of NLP) actually boosts performance.
- Sampling negative class documents does degrade evaluation performance, while it improves the efficiency of classification and learning.
- Using all available terms gives the best performance, though sampling terms does not

degrade performance substantially, while it improves the efficiency of classification and learning.

- The kernel-Adatron algorithm gives a very reasonable performance, though it is a much simpler algorithm that the examined SMO variations.
- Using cross-validation for customizing the parameters of learning should improve the quality of the learnt SVM classifiers.
- Our simple thresholding results are encouraging. Using a principled approach to thresholding (such as beta-gamma or other distribution based approaches) may prove practical and effective.

5. Concluding Thoughts

As the additional analyses in Figures 6–9 show, our experiments on the TREC Assessor topics underscore the comparative strengths and differences in the two filter types we have developed. For IR-based filters, we see responsiveness and delivery of relevant documents even when there are limited training data. They also were very fast to train and run and generally required only a handful of features (cf. Figure 10). The IR-based filters failed to return relevant documents in only one case out of fifty topics. However, a measure that rewards the delivery of *no* documents, such as T11SU, penalizes the IR-based approach. In contrast, we see consistent positive (or neutral) performance from SVM-based filters, giving high precision, if under-delivery, on unseen data. However, for many of the SVM runs (on on more than twenty topics) there were no documents returned at all.

The challenge in many practical (commercial) applications is limited training data and the need to optimize performance in virtually real time.
Some of the best methods for classifier training, such as kernel-based approaches, require significant amounts of data and may depend on sensitive parameter tuning. However, we see in our own experiments that kernel methods can give high precision and accuracy. If we can overcome their high-precision bias using thresholding or uneven-margin-based learning and adapt them to sparse data, they may become an attractive solution. We also see the robustness and generally good performance of IR-based approaches. Perhaps the ideal application will combine features of both and optimize the choice of classifier—IR or SVM—for each topic on a case by case basis

convict child rapist (40.4)	child rapist (39.6)
convict child rapist marc dutroux (39)	eefje (38.1)
rapist marc dutroux lambrecks (37.6)	marchal (38.1)
child rapist marc dutroux (37.6)	lambrecks (37.6)
eefje lambrecks (37)	dardenne (37.3)

Figure 10. CLARIT terms/weights for topic 103

References

[1] Anlauf JK, Biehl M, The adatron: an adaptive perceptron algorithm. *Europhys. Letters*, 10, 1989, 687–692.

[2] Boser BE, Guyon IM, Vapnik VN, A training algorithm for optimal margin classifiers. In D. Haussler, editor, *5th Annual ACM Workshop on COLT.* Pittsburgh, PA: ACM Press, 1992, 144–152.

[3] Cancedda N, Cesa-Bianchi N, Conconi A, Gentile C, Goutte C, Li Y, Renders JM, Shawe-Taylor J, Vinokourov A, Kernel Methods for Document Filtering. *TREC 2002 Notebook Papers*, 2002.

[4] Cristianini N, Lodhi H, Shawe-TaylorJ, Latent Semantic Kernels. *Journal of Intelligent Information Systems* (JJIS) Vol. 18, No. 2, 2002.

[5] Evans DA, Shanahan JG, Tong X, Roma N, Stoica E, Sheftel V, Montogomery J, Bennett J, Fujita S, Grefenstette G, Topic-Specific Optimization and Structuring. A Report on CLARIT TREC-2001 Experiments. In EM Voorhees and DK Harman (Editors), *The Tenth Text REtrieval Conference (TREC-2001).* NIST Special Publication 500-250. Washington, DC: U.S. Government Printing Office, 2002, 132–141.

[6] Frieß T-T, Cristianini N, Campbell C, The kernel adatron algorithm: A fast and simple learning procedure for support vector machines. *15th Intl. Conf. Machine Learning.* Morgan Kaufmann Publishers, 1998.

[7] Keerthi SS, Shevade SK, Bhattacharyya C, Murthy KRK, *Improvements to Platt's SMO algorithm for SVM classifier design.* Technical report, Dept of CSA, IISc, Bangalore, India, 1999.

[8] Lee K-S, Oh J-H, Huang JX, Kim J-H, Choi K-S, TREC-9 Experiments at KAIST: QA, CLIR and Batch Filtering. *TREC Proceedings 2000*, 300*ff*.

[9] Lodhi H, Shawe-Taylor J, Christianini N, Watkins C, Text classication using string kernels. *Advances in Neural Information Processing Systems*, 13, 2001.

[10] Platt J, Fast Training of Support Vector Machines using Sequential Minimal Optimization, in *Advances in Kernel Methods - Support Vector Learning*, Schölkopf B, Burges C, Smola A, eds., MIT Press, 1998.

[11] Santamaria J, Pantaleon C, Principe JC, Minimising BER in DFE with the Adatron Algorithm, *Neural Networks for Signal Processing XI (NNSP 2001)*, Falmouth, MA, 2001, 423–432.

[12] Vapnik V, *The Nature of Statistical Learning Theory.* New York, NY: Springer, 1995.

[13] Zhai C, Jansen P, Stoica E, Grot N, Evans DA, Threshold Calibration in CLARIT Adaptive Filtering. In EM Voorhees and DK Harman (Editors), *The Seventh Text REtrieval Conference (TREC-7).* NIST Special Publication 500-242. Washington, DC: U.S. Government Printing Office, 1999, 149–156.

[14] Zhai C, Jansen P, Roma N, Stoica E, Evans DA., Optimization in CLARIT TREC-8 Adaptive Filtering. In EM Voorhees and DK Harman (Editors). *The Eighth Text REtrieval Conference (TREC-8).* NIST Special Publication 500-246. Washington, DC: U.S. Government Printing Office, 2000, 253–258.

CLIPS at TREC-11: Experiments in Filtering

Christophe Brouard
Laboratoire CLIPS
Equipe MRIM
BP. 53
38041 Grenoble cedex 9, France
Christophe.Brouard@imag.fr

Abstract

At the TREC9 conference, we presented a new adaptive filtering system called RELIEFS. This system which is based on the idea of resonance, combines for each term t, the relative frequency of relevance knowing t and the relative frequency of t kwowing relevance. On the basis of other experiments, several changes have been made. We improved our threshold adaption, we slightly changed our relevance evaluation function and we gave up the use of conjunctions and thesaurus. The system is now focusing more exclusively on the combination of both reverse frequencies that we believe to represent the fundamental aspects of relevance estimation. This year we used the system in its new version and we tested it on the Reuters corpus. Focusing on the combination of the two frequencies, we varied their relative importance. The results show globally a good effectiveness especially when both frequencies are balanced.

1. Introduction

In a logical approach, it has been found that the evaluation of relevance of a document D to a query Q can be based on the evaluation of the implication from document to query D->Q (van Rijsbergen, 1986). Some authors also emphasized the role of the reverse implication (Q->D) (Nie, 1988). These two entailment relations can also be found in most probabilistic models, formulated as P(Q/D) (Probability of Q knowing D) and P(D/Q) (see Crestani, Lalmas, van Rijsbergen & Campbell, 1998, for an overview of Probabilistic models).

If one considers a document as a set of terms, and a query as a specification of what we are looking for, the entailment D->Q may be decomposed to the judgment of "if the term is present then the document is relevant" for each term of the document. So, if we use a set of documents with the relevant judgments associated, the entailment D->Q can be decomposed in a set of relative frequencies of relevance knowing term for each term of the document. Similarly, the reverse entailment relation can be decomposed in a set of relative frequencies of term knowing relevance. From a pragmatic point of view, the use of the first frequency is easily understood since it allows to favour the terms which predict relevance (when the term is present then the document in which it occurs is relevant). The consideration of the reverse frequencies may be justified by the fact that it allows to favour terms which have been met many times in relevant documents. Since they appear relatively frequently, these terms are likely to be usefull for the next evaluations, and we can think that their presence in relevant documents is not a coincidence (comparing to rare terms which occured once or twice).

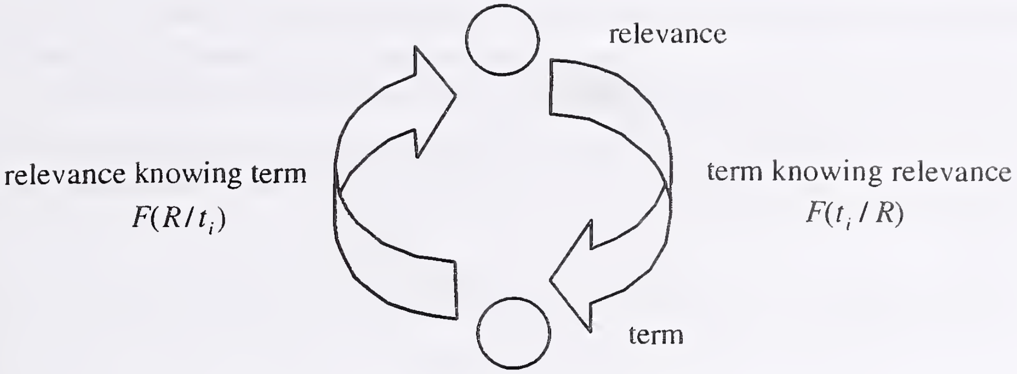


Figure 1 – Relationships between terms and relevance

We believe that both entailment relations capture the essence of relevance. As we noted before they are found in the most IR models but not always in a obvious way. With the system RELIEFS we try to isolate them and to use them in a simple way. So, for each term t_i, we propose to compute the frequency of relevance knowing term $F(R/t_i)$ and the reverse frequency $F(t_i/R)$ (Figure 1).

2. System description

The RELIEFS document processing can be decomposed in three steps:

1. Selection of N terms from the document,
2. Estimation of the document's relevance,
3. Revision of term's relative frequencies.

2.1 Step 1 : Selection of N document terms

All the document terms are compared with the terms which have been extracted from the query, from the document examples given for learning and from the documents which have been previously selected. They are sorted by the value the product $F(R/t_i)*F(t_i/R)$. If less than N terms can be selected in this way (in our experiments we choose N=30), this selection is completed by the document terms in their lecture order.

2.2 Step 2 : Relevance estimation

Considering the terms which appear in the document, the score of the document is computed as follows :

$$\frac{\sum_{i=1}^{N} F(R/t_i)*F(t_i/R)}{\sum_{j=1}^{N} F(R/t_j)*F(t_j/R)}$$

where i are the indices of the N best terms (according to the product $F(R/t)*F(t/R)$)) which are present in the document and where j are the indices of the N best terms (present or not present in the document). In RELIEFS, the relevance of a document can be interpreted as a kind of resonance in a network (Figure 2) in which each relative frequency corresponds to a weighted connection and each term corresponds to a node (Brouard & Nie, 2003). A relevance node is built for each query.

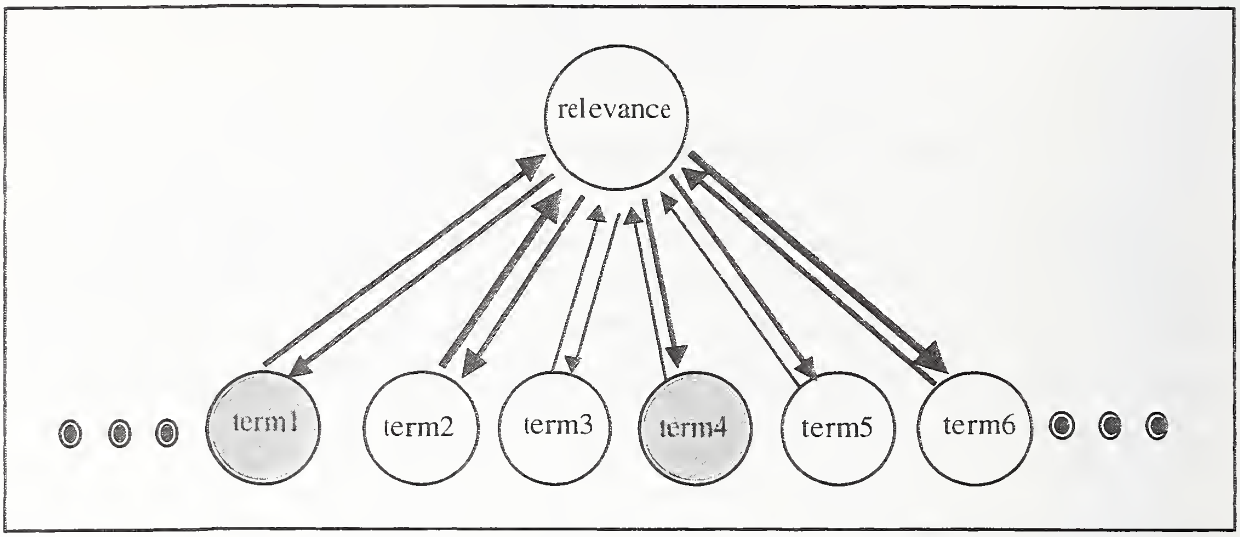


Figure 2 - Relevance evaluation in RELIEFS of a document in which term1 and term4 occur

2.3 Step 3 : Relative frequencies updating

If the evaluation of step2 is larger than a defined threshold the document is selected and the relevance feedback takes place. Given the relevant judgment, the N terms selected in the first step are submitted to an updating process on their relative frequencies.

2.4 Threshold adaption

The queries are very different from each other. Therefore, a different threshold is determined for each query. The value of the threshold is determined empirically. The initial threshold is computed on the basis of the score of the three first relevant documents. We used the following strategy to adjust the value:

- If a selected document is irrelevant, the system is considered to be too tolerant. The threshold is increased.

- If a document (that we do not know if it is relevant or not) is not selected, the system is considered to be too strict. Then the threshold is decreased

The increase scale is set to be higher than the decrease scale because there are much more unselected documents than the selected ones. The decrease of threshold in the second case is due to the fact that we do not want the system to remain silent for a too long period. This allows to gradually correct an initial threshold that is fixed too high. In both cases, we considered different criteria for the modification of the threshold, including:

- The number of irrelevant documents that are selected consecutively: The higher this number, the larger the increase scale and the smaller the decrease scale.

- The number of consecutive relevant document: The higher this number, the larger the decrease scale. This change only concerns the decrease case.

- The number of documents considered: The larger this number, the lower the change scales. The intuition behind this criterion is that we would make larger changes at the beginning of the filtering. When a certain number of documents have been treated, the system should stabilize.

We also considered a more global criterion on the probability of relevance. Each score is associated with a frequency of relevance and we try to adjust the threshold on the score for which the probability is 0.33 (this selection criterion is optimal for the utility measure).

3. Experiments

3.1 Preprocessing

We used the Porter's stemmer to get the root form of every term and we removed the stopwords.

3.2 Details of processing

All parameters (which mainly concern the threshold tuning) were tuned on the Oshumed corpus (TREC9). For each document of the Reuters corpus, we only considered the title and the text fields. For updating frequencies, the unjudged documents were considered as irrelevant. A pseudo-relevance feedback has been applied when the scores were smaller than a defined threshold. In this case, the document is considered as irrelevant.

3.3 Varying the relative importance of the entailment relations

In our previous tests, both entailment relations are considered with the same importance: they are multiplied. Our question is, should they play equal role in the estimation or should one factor be more important than another? In order to answer this question, we vary the importance of one of the implication by replacing the weight $F(R/t_i)$ by $F(R/t_i)^\rho$. The factor ρ changes the relative importance of this relation. The relevance estimate is defined as follows:

$$\frac{\sum_{i=1}^{N} F(R/t_i) * F(t_i/R)^\rho}{\sum_{j=1}^{N} F(R/t_j) * F(t_j/R)^\rho}$$

3.4 Results

Globally, it turns out that the best results are obtained when the two frequencies are balanced (ρ in [1.0,1.4]) even if we can observe that good results for $\rho = 1.6$ and $\rho = 1.7$ (Table 1). This is consistent with the previous observations we obtained with the Oshumed corpus.

ρ	Utility	ρ	Utility	ρ	Utility	ρ	Utility
0.0	-0.26	0.5	10.35	1.0	16.68	1.5	14.01
0.1	2.65	0.6	12.25	1.1	17.25	1.6	16.82
0.2	4.92	0.7	12.42	1.2	17.84	1.7	17.27
0.3	8.01	0.8	13.22	1.3	17.68	1.8	13.19
0.4	9.41	0.9	15.44	1.4	16.95	1.9	12.03

Table 1 : The impact of ρ on the utility.

We submitted the best run ($\rho = 1.2$) for comparison on utility criteria. The comparison is favourable since about 80% of the scores are above or equal to the median. Like the other systems we obtained better results for the first 50 queries than for the 50 last ones which are built differently.

In conclusion, results suggest that the two reverses frequencies are sufficient clues to estimate the relevance. Furthermore, the fact that the best results are obtained when the two aspects are balanced should indicate that they are both necessary and equally important.

References

Brouard, C., & Nie, J.-Y. (2000, 11/00). The system RELIEFS: a new approach for information filtering. Text Retrieval Conference (TREC-9), Washington D.C.

Brouard, C., & Nie, J.Y. (2003). Relevance as Resonance : A new Theoretical Perspective and a Practical Utilisation in Information Filtering. *Information Processing & Management*. Revision under review.

Crestani, F., Lalmas, M., C.J, V. R., & Campbell, I. (1998). Is This Document Relevant?...Probably: A Survey of Probabilistic Models in Information Retrieval. *ACM Computing Surveys*, 30(4), 528-552.

Nie, J. Y. (1988). An outline of a general model for information retrieval. Proceedings of the 11th Annual ACM Conference on Research and Development in Information Retrieval, Grenoble.

van Rijsbergen, C. J. (1986). A non-classical logic for information retrieval. *The Computer Journal*, 29(6), 481-485.

CLIPS at TREC-11: Experiments in Video Retrieval

Georges M. Quénot, Daniel Moraru, Laurent Besacier and Philippe Mulhem

CLIPS-IMAG, BP53, 38041 Grenoble Cedex 9, France
Georges.Quenot@imag.fr

Abstract

This paper presents the systems used by CLIPS-IMAG to perform the Shot Boundary Detection (SBD) task, the Feature Extraction (FE) and the Search (S) task of the Video track of the TREC-11 conference. Results obtained for the TREC-11 evaluation are presented.

1 Introduction

The CLIPS-IMAG laboratory has participated to all of the three tasks proposed in the video track of the TREC-11 evaluation. This participation was done in collaboration with teams from other institutions including LIMSI-CNRS (Orsay, France) for speech transcription, LIT-IPAL (Singapore) for face detection and INSA (Lyon, France) for text transcription. The following sections describe our participation to the tasks.

2 Shot Boundary Detection Task

The system used by CLIPS-IMAG to perform the TREC-11 SBD task is almost the same as the one used for the TREC-10 evaluation [1]. This system detects "cut" transitions by direct image comparison after motion compensation and "dissolve" transitions by comparing the norms of the first and second temporal derivatives of the images. It also has a special module for detecting photographic flashes and filtering them as erroneous "cuts". With respect to the system used for the TREC-10 evaluation, this one has an additional module for detecting additional "cuts" via a motion peak detector. Some parameters controlling the existing modules have been tuned using the TREC-10 SBD corpus and reference segmentation, and a global parameter for the tuning of the recall versus precision compromise has been inserted. The system is still globally organized according to a (software) dataflow approach and Figure 1 shows its architecture.

The original version of this system was evaluated using the INA corpus and the standard protocol [2] (http://asim.lip6.fr/AIM/corpus/aim1/indexE.html) developed in the context of the GT10 working group on multimedia indexing of the ISIS French research group on images and signal processing. The TREC-10 and TREC-11 SBD tasks partly reused this test protocol (with different test corpora). The reference segmentation for the search, the feature test and the feature search collections of the TREC-11 corpus were also built with this system (the version used for the TREC-10 evaluation).

2.1 Cut detection by Image Comparison after Motion Compensation

This system was originally designed in order to evaluate the interest of using image comparison with motion compensation for video segmentation. It has been complemented afterward with a photographic flash detector and a dissolve detector.

2.1.1 Image Difference with Motion Compensation

Direct image difference is the simplest way for comparing two images and then to detect discontinuities (cuts) in video documents. Such difference however is very sensitive to intensity variation and to motion. This is why an image difference after motion compensation (and also gain and offset compensation) has been used here.

Motion compensation is performed using an optical flow technique [3] which is able to align both images over an intermediate one. This particular technique has the advantage to provide a high quality, dense, global and continuous matching between the images. Once the images have been optimally aligned, a global difference with gain and offset compensation is computed.

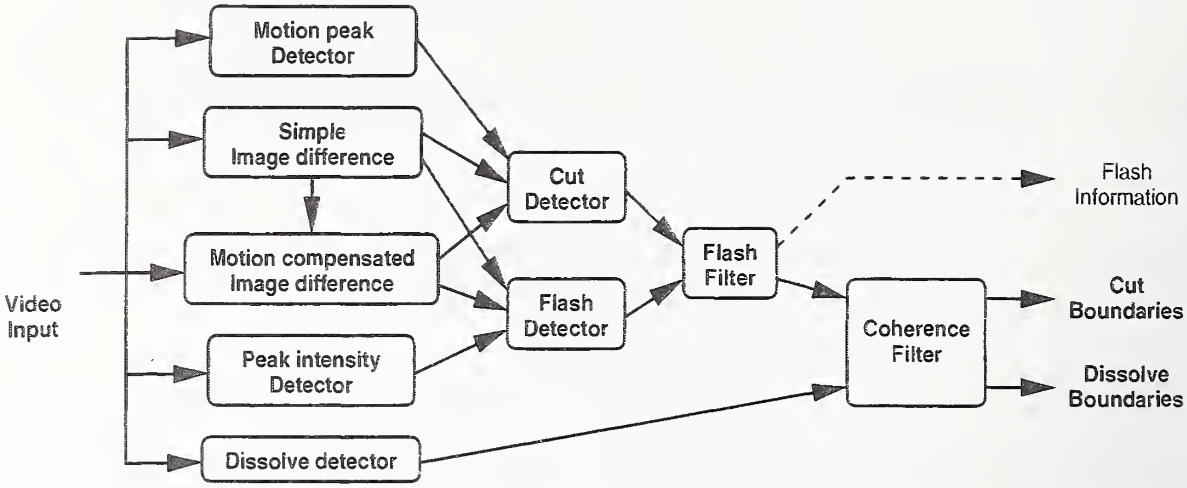


Figure 1: Shot boundary detection system architecture

Since the image alignment computation is rather costly, it is actually computed only if the simple image difference with gain and offset compensation alone has a high enough value (i.e. only if there is significant motion within the scene). Also, in order to reduce the computation cost, the differences (with and without motion compensation) are computed on reduced size images (typically 96 × 72 for the PAL video format). A possible cut is detected if both the direct and the motion compensated differences are above an adaptive threshold.

In order for the system to be able to find shot continuity despite photographic flashes, the direct and motion compensated image difference modules does not only compare consecutive frames but also, if needed, frames separated by one or two intermediate frames.

2.1.2 Photographic flash detection

A photographic flash detector feature was implemented in the system since flashes are very frequent in TV news (for which this system was originally designed for) and they induce many segmentation errors. Flash detection has also an interest apart from the segmentation problem since shots with high flash density indicates a specific type of event which is an interesting semantic information.

The flash detection is based on an intensity peak detector which identify 1- or 2-frame long peaks of the average image intensity and a filter which uses this information as well as the output of the image difference computation modules. A 1- or 2-frame long flash is detected if there is a corresponding intensity peak and if the direct or motion compensated difference between the previous and following frames are below a given threshold. Flash information may be output toward another destination. In the segmentation system, it is used for filtering the detected "cut" transitions.

2.2 Dissolve detection

Dissolve effects are the only continuous transition effects detected by this system. The method is very simple: a dissolve effect is detected if the L_1 norm (Minkowski distance with exponent 1) of the first image derivative is high enough compared to the L_1 norm of the second image derivative (this checks that the pixel intensities roughly follows a linear but non constant function of the frame number). This actually detects only dissolve effects between constant or slowly moving shots. This first criterion is computed in the neighborhood (± 5 frames) of each frame and a filter is then applied (the effect must be detected or almost detected in several consecutive frames).

2.3 Output filtering

A final step enforces consistency between the output of the cut and dissolve detectors according to specific rules. For instance, if a cut is detected within a dissolve, depending upon the length of the dissolve and the location of the cut within it, it may be decided either to keep only one of them or to keep both but moving one extremity of the dissolve so that it occurs completely before or after the cut.

2.4 New features

2.4.1 Motion peak detection

The main new feature of the system is the motion peak detection module. It was observed from TREC-10 and other evaluations that the motion compensated image difference was generally a good indicator of a "cut" transition but, sometimes, the motion compensation was too good at compensating image differences (and even more when associated to a gain and offset compensation) and quite a few actual "cuts" were removed because the pre- and post-transition images were accidentally too close after motion compensation. We found that it is possible not to remove most of them because such compensation usually requires compensation with a large and highly distorted motion wich is not present in the previous and following image-to-image change. A "cut" detected from simple image difference is then removed if it is not confirmed by motion compensated image difference *unless* it also corresponds to a peak in motion intensity.

2.4.2 Global tuning parameter

The system has several thresholds that have to be tuned for an accurate detection. Depending upon their values, the result can detect or miss more transitions. These thresholds also have to be well balanced among themselves to produce a consistent result. Most of them were manually tuned as the system was built in order to produce the best possible results using sample data. No additional tuning was done for the TREC-10 evaluation. A first run was made using the default system threshold (originally oriented toward a high recall) and a second run with lower thresholds (20 % lower) in order to further improve the recall.

For the TREC-11 evaluation, as well as for other applications of the system, we decided to have all the threshold parameters be a function of a global parameter controlling the recall versus precision compromise (or, more precisely, the false positive to false negative ratio). A function was heuristically devised for all of them. A power low has been chosen. A first system tuning was done using the TREC-10 SBD corpus and reference segmentation in order to set a point at which the false positives are roughly equivalent to the false negatives. Then a power coefficient has also been tuned for each parameter in order to have the ratio to follow also roughly a power law.

2.5 Evaluation using the TREC-11 SBD test data

Ten runs have been submitted for the CLIPS-IMAG system. These correspond to the same system with a variation of the global parameter controlling the recall versus precision compromise. This parameter has been varied so that the target false positive to false negative ratio has extreme values of roughly 3:1 and 1:3 with intermediate ones following roughly a power law.

As expected, this made possible the drawing of a recall × precision curve. Figure 2 shows these curves for the features selected for the evaluation. There are three recall × precision curves respectively for all transitions, for cut transitions and for gradual transitions. There is also a frame-recall × frame-precision curve that qualifies the accuracy of the boundaries of recovered gradual transitions. For comparison purposes, the results of other systems are plotted as set of points (with abbreviated names given with the results by NIST).

The CLIPS system appears to be very good for gradual transitions both for the detection and the location. This may come from the specificity of TREC-11 video data which are quite old and which mostly contain dissolve or fade gradual transitions (other special effects were not common in the forties/fifties). This is the only type of gradual effect our system was designed for. This indicates also that the chosen method (comparison of the first and second temporal derivative of the images) is quite good even if theoretically suited only for sequences with no or very little motion.

The CLIPS system appears to be in the average for cut detection but thanks to its very good performance in gradual transition detection and considering that these are more difficult to detect than cuts, its global performance for all transitions also remains very good.

3 Feature Search Task

CLIPS extracted only features 3 (faces), 4 (people), 8 (speech) and 10 (monologue).

3.1 Face and People Detection

Face and people detection were based on a face detection tool available from CMU [4]. This tool was run (by Philippe Mulhem and colleagues at Laboratories for Information Technologies, Singapore) on one keyframe automatically extracted for each shot. The keyframe was selected within the shot simply as the one having the highest contrast (in order to avoid frames within

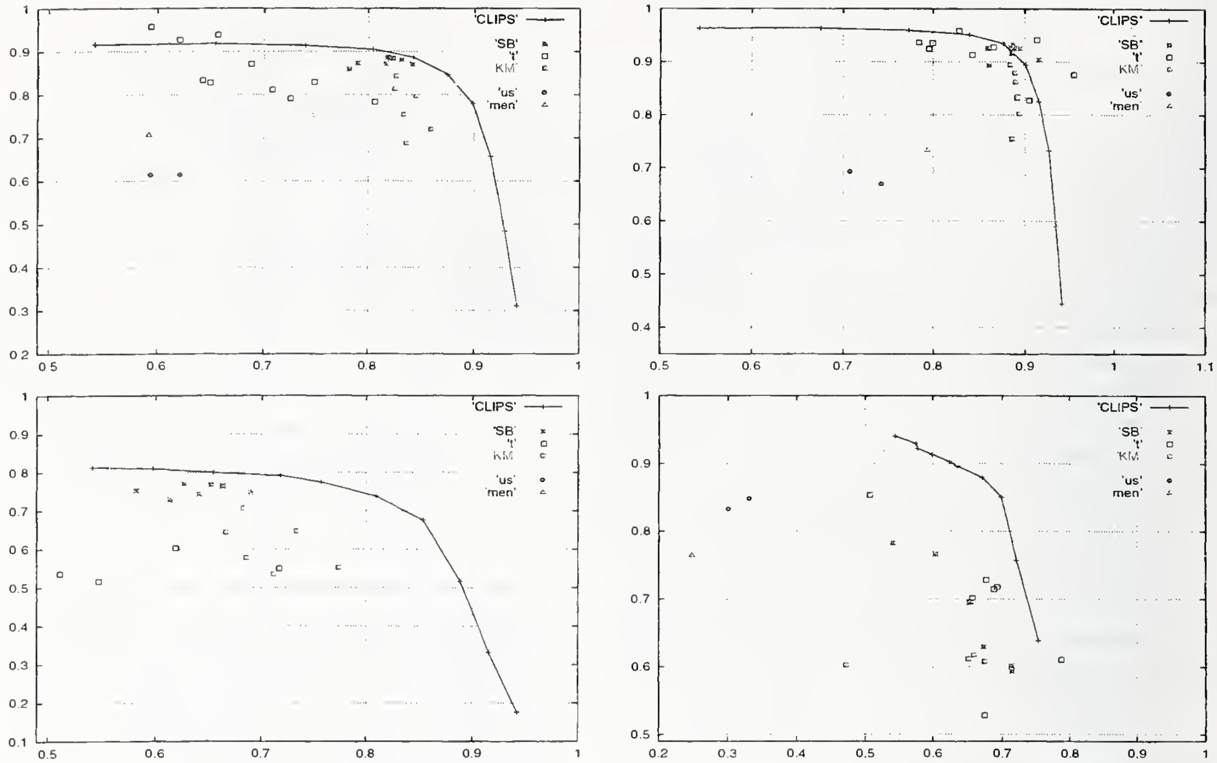


Figure 2: Recall × Precision global results for all (top left), cut (top right) and gradual (bot. left) transitions; Frame-Recall × Frame-Precision global results for gradual transitions (bot. right).

fades and dissolves). People were only detected on the basis of the presence of at least two faces. The results were ranked according to the presence of one (or at least two) face(s) and to the total face area.

Table 1 and 2 show the performance of the CLIPS system among other systems that have searched for features 3 and 4. The quality is quite low for these features. This comes probably from the simplicity of the approach only based on keyframe extraction followed by face detection (which is by itself quite good however), especially for people detection.

3.2 Speech and Monologue Detection

3.2.1 Speech Feature Detection

Both for speech and monologue feature detection, the acoustic vectors extracted from speech were conventional parameters used in speech processing, i.e. 16 MFCC (Mel Frequency Cepstral Coefficients) and their log energy computed every 10ms on 20 ms signal windows with no Cepstral Mean Subtraction (CMS) applied.

At first, we eliminated the silent films using the energy calculation of the signal. Then, the idea (Figure 3) was to train a speech model and a non-speech model (also called world model) and to compute the log-likelihood ratio between both models. We used GMMs (Gaussian Mixtures Models) to characterize speech and non-speech. The GMMs were made of 128 gaussian components and trained using the ELISA platfrom [5].

Suppose we have speech model M_{Spch}, a world model M_{UnSpch} and a acoustic vector sequence $X = x_1 \dots x_n$. The log-likelihood ratio between the hypothesis of X being speech and not being speech is defined by:

$$llr(X) = \log P(X/M_{Spch}) - \log P(X/M_{UnSpch})$$

The bigger the ratio is the bigger the probability of X being speech is.

The speech model M_{Spch} was trained on about 2.5 hours of speech manually selected from the DEV files. The world model M_{UnSpch} was trained on everything that was left from the DEV files (about 2.5 hours). The log-likelihood ratio was computed for every shot and the results were sorted descendant.

rank	system	A.P.	D.100	D.1000	rank	system	A.P.	D.100	D.1000
1	B_r1_1	0.613	99	303	6	B_E2002_1	0.154	53	114
2	B_RA_1	0.473	86	253	7	B_om1_1	0.150	28	255
3	B_M-1_1	0.327	51	312	8	B_Sys1_1	0.111	17	190
4	B_M-2_2	0.288	53	293	9	B_I2_2	0.091	56	57
5	**CLIPS**	**0.178**	**70**	**118**	10	B_I1_1	0.089	55	55

Table 1: Average precision and average hits at depth 100 and 1000 for feature 3 (faces).

rank	system	A.P.	D.100	D.1000	rank	system	A.P.	D.100	D.1000
1	A_r2_2	0.274	57	277	6	B_r1_1	0.050	45	48
2	B_M-1_1	0.271	31	361	**7**	**CLIPS**	**0.023**	**18**	**18**
3	B_T1_1	0.248	54	251	8	B_I1_1	0.008	12	12
4	B_T2_2	0.168	27	223	9	B_I2_2	0.008	10	10
5	B_Sys1_1	0.071	44	83					

Table 2: Average precision and average hits at depth 100 and 1000 for feature 4 (people).

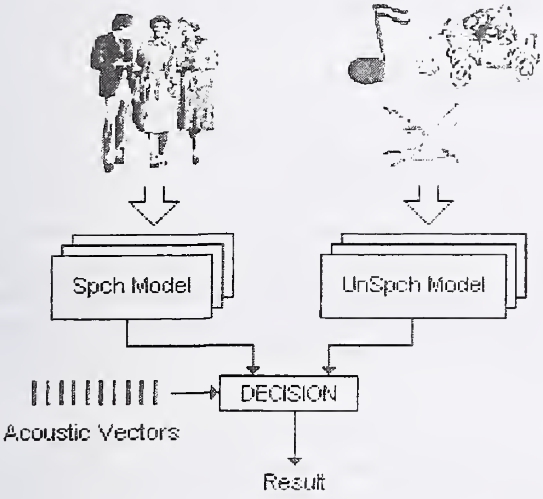


Figure 3: Speech Feature Detection

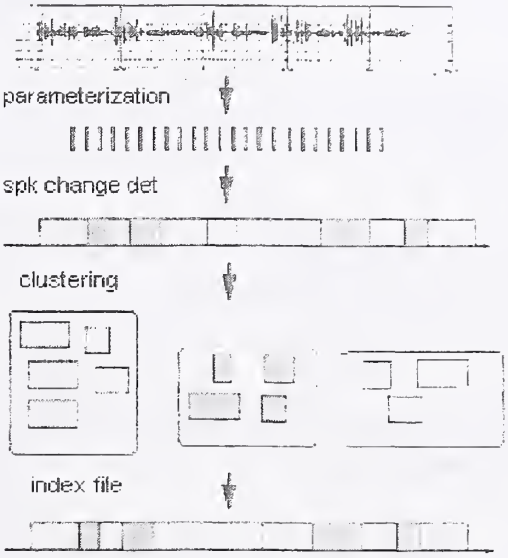


Figure 4: Speaker Segmentation System

3.2.2 Monologue Feature Detection

For the monologue feature detection we used the CLIPS Segmentation System [6] used during last NIST 2002 Speaker Recognition Evaluation, combined with the results from the speech feature detection task. The CLIPS Speaker Segmentation System is presented in Figure 4.

Once the speech is parameterized the segmentation is done in two steps. At first the speaker change points are detected using the Bayesian Information Criterion [7]. The purpose is to cut the file in single-speaker segments. Then the segments are grouped by speakers using an hierarchical clustering algorithm. At the end an index file is created containing all the speaker information obtained.

In order to perform monologue detection only on speech segments, the speaker segmentation system was applied only on the TEST files that had at least one shot in the top 300 of the speech feature detection task. Then, the shots labeled as monologue shots where the shots which were found to contain only one speaker for their whole duration (Figure 5).

The selected shots were finally sorted by the log-likelihood ratio computed during the speech feature detection task.

rank	system	A.P.	D.100	D.1000	rank	system	A.P.	D.100	D.1000
1	**CL-LIMSI**	**0.721**	**100**	**997**	8	B_T1_1	0.645	95	934
2	B_M-1_1	0.713	99	990	9	B_T2_2	0.645	95	934
3	B_E2002_1	0.710	100	987	10	B_Sys1_1	0.645	97	932
4	B_I1_1	0.681	96	970	11	B_r1_1	0.642	92	936
5	B_I2_2	0.681	96	970	12	A_r2_2	0.630	95	924
6	B_Sys2_2	0.663	98	951	13	B_RA_1	0.570	100	792
7	**CL-GEOD**	**0.649**	**98**	**924**					

Table 3: Average precision and average hits at depth 100 and 1000 for feature 8 (speech).

rank	system	A.P.	D.100	D.1000	rank	system	A.P.	D.100	D.1000
1	B_M-1_1	0.268	14	37	6	B_I2_2	0.009	1	1
2	**CL-LIMSI**	**0.149**	**23**	**23**	7	B_RA_1	0.009	0	16
3	**CL-GEOD**	**0.117**	14	14	8	B_Sys2_2	0.009	1	14
4	B_r1_1	0.082	13	16	9	B_Sys1_1	0.008	1	14
5	B_I1_1	0.009	1	1					

Table 4: Average precision and average hits at depth 100 and 1000 for feature 10 (monologue).

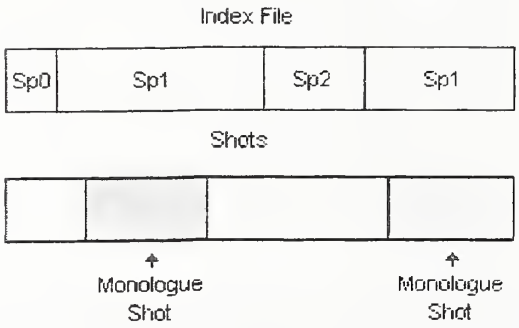


Figure 5: Monologue Feature Detection using Speaker Segmentation index file

3.2.3 Speech and Monolgue Features Evaluation

Alternatively to the above described system, we also used the output of the LIMSI Audio-Video transcription system [8]. This system is the one used for the LIMSI donated transcription for which we additionally had a speaker segmentation. The ranking was done using the same principles.

Table 3 and 4 show the performance of CLIPS-LIMSI and CLIPS-GEOD (described in sections 3.2.1 and 3.2.2) systems among other systems that have searched for features 8 and 10. The quality is very good for all systems for speech detection. LIMSI is ranked first and GEOD is in the average. The monologue detection is more selective and CLIPS-LIMSI and CLIPS-GEOD are ranked respectively 2 and 3 probably due to a good face detection.

4 Search Task

CLIPS-IMAG submitted three runs for the search task. One is based only on speech transcription (from LIMSI-CNRS), one based only on a combination of donated features, and one based on a combination of both. We did not use anything else like image similarity for instance.

A vectorial model was used both for the keyword-based search, for the combination of donated features, and for the combination of keywords and features. A weight can be given independently to each keyword (stemming was used) and to each donated feature. Independently weight can be given to the keyword based search and to the feature based search. A single system is used for the three runs. For the "ASR only" , the "ASR+features", and the "features only" runs, the keywords/features weights are respectively set to (1,0), (0.5,0.5) and (0,1). The selected keywords and features as well as their relative weight are chosen manually and once for the three runs.

Our three runs were manual only and of type A. However, the only use that we have made of the test corpus is an evaluation of the quality of the donated features (all of type B) in order to weight them accordingly. There is a fixed weighting of the donators for each feature according to a quality evaluation (which is combined to the weight of the features and to the keywords/features weights). Since the feature quality

rank	system	A.P.	D.10	D.100	rank	system	A.P.	D.10	D.100
1	M_B_ci_1	0.231	6.360	10.880	12	M_B_MT1_2	0.034	1.520	3.560
2	M_B_M-2_2	0.136	2.720	10.240	13	M_B_Aqt_3	0.026	0.480	3.600
3	M_B_UAL1_1	0.112	2.440	9.200	14	M_A_UAL2_4	0.026	0.320	4.920
4	M_B_M-3_3	0.093	2.240	9.160	15	M_B_MT2_3	0.019	0.880	2.280
5	M_B_0_T_2	0.092	1.920	7.240	16	M_B_eo.3_1	0.010	1.000	2.400
6	**CLIPS-ASR**	**0.071**	**1.560**	**7.240**	17	M_B_M-1_1	0.006	0.400	2.560
7	**CLIPS-A+F**	**0.064**	**1.520**	**3.840**	18	M_B_0_TIscG_4	0.004	0.120	1.040
8	M_B_KM-2_2	0.060	1.280	5.520	**19**	**CLIPS-Feat.**	**0.003**	**0.240**	**1.600**
9	M_B_qtrec_2	0.059	1.520	6.840	20	M_B_0_TIsc_3	0.002	0.080	1.400
10	M_B_KM-4_4	0.057	1.720	5.280	21	M_B_0_TIac_1	0.002	0.040	1.200
11	M_B_KM-3_3	0.043	1.160	5.320					

Table 5: Average precision and average hits at depth 10 and 100 for systems ran manually for the search task.

evaluation is the only use that we have made of the test corpus ans since we do not expect this quality evaluation to be very sensitive to this, our runs are almost of type B runs and we consider that the comparison with type B runs is meaningful.

Table 5 shows the performance of CLIPS systems among other systems that have processed manually all the 25 topics. Our "ASR only" and "ASR+features" runs ranked respectively 6 and 7 (on average precision) while the "features only" run ranked 19. Even though the topics were chosen in order not to favour speech recognition, the "ASR only" system performed slightly better than the "ASR+features" system. The feature only result is very poor probably because for many topics they are not very discriminative or even relevant.

5 Conclusion

We have presented the participation of the CLIPS-IMAG laboratory to the video track of the TREC-11 evaluation. We participated in all of the three proposed tasks. This participation was done in collaboration with teams from other institutions including LIMSI-CNRS (Orsay, France) for speech transcription, LIT-IPAL (Singapore) for face detection and INSA (Lyon, France) for text transcription. Our performance was quite good in shot boundary detection, average or poor for face and people detection, good for speech and monologue detection and quite good for the search task with speech recognition and poor without it.

References

[1] Quénot, G.M.,: TREC-10 Shot Boundary Detection Task: CLIPS System Description and Evaluation, In em 10th Text Retrieval Conference, Gaithersburg, MD, USA, 13-16 November, 2001.

[2] Ruiloba, R., Joly, P., Marchand, S., Quénot, G.M.: Toward a Standard Protocol for the Evaluation of Temporal Video Segmentation Algorithms, In *Content Based Multimedia Indexing*, Toulouse, Oct. 1999.

[3] Quénot, G.M.: Computation of Optical Flow Using Dynamic Programming, In *IAPR Workshop on Machine Vision Applications*, pages 249-52, Tokyo, Japan, 12-14 nov. 1996.

[4] Rowley, H., Baluja, S., Kanade, T.: Neural Network-Based Face Detection, In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 20, number 1, pages 23-38, January 1998.

[5] Magrin-Chagnolleau, I., Gravier, G, Blouet, R. for the ELISA consortium: Overview of the 2000-2001 ELISA consortium research activities, In *2001: A Speaker Odyssey*, pp.6772, Chania, Crete, June 2001.

[6] Moraru, D., Meignier, S.,Besacier, L., Bonastre, J.-F., Magrin-Chagnolleau, Y.: The ELISA Consortium Approaches in Speaker Segmentation during The NIST 2002 Speaker Recognition Evaluation In *Proceedings of ICASSP*, Hong Kong, 6-10 apr. 2003.

[7] Delacourt, P., Wellekens, C.: DISTBIC: a speaker-based segmentation for audio data indexing, In *Speech Communication*, Vol. 32, No. 1-2, September 2000.

[8] Barras, C., Allauzen, A., Lamel, L., and Gauvain, JL,: Transcribing Audio-Video Archives, In *Proceedings of ICASSP*, pages 13-16, Orlando, May 2002.

Experiments in Novelty Detection at Columbia University

Barry Schiffman
Department of Computer Science
Columbia University
bschiff@cs.columbia.edu

Abstract

This paper describes the method we used for the Novelty Track for the 2002 Text Retrieval Conference (TREC). We tried to adapt tools we are developing for a task closely related to the novelty part of the this track. The system we are building will scan a stream of documents and present to the user only the new information it finds. For the "relevance" part of the TREC, we decided to test the applicability of some of these tools. Since information retrieval is not a focus of our research, we thought it would be more interesting to use something new rather than try to hurriedly catch up. The results were far from satisfactory, but it is clear from the overall results that novelty detection remains a difficult and unsolved problem.

1 Introduction

The task in the Novelty Track at the 2002 Text Retrieval Conference (TREC) was structured in two parts. First, the system had to find sentences in a cluster of documents that are relevant to a query, and second, as the sentences were presented in a predetermined order, it had to remove any that duplicated information in previous sentences. The clusters themselves were culled from the fourth and fifth TREC collections by an Information Retrieval system, selecting the documents relevant to the query. The queries were 50 previous TREC topics, in some cases altered somewhat. Up to 25 documents were collected for each topic.

Our interest in participating in the Novelty Track was to work with the data in the second part. We are building a system, called the New Information Agent (NIA) to detect new information from a stream of document. Like the TREC version, the input to our system is a clustered stream of documents, or in an offline version, a collection of documents, that focuses on a particular event or issue. Again, like the TREC task, the output of our system is a short list of the sentences that do not contain any material that duplicates a passage selected earlier. But the presence of the query is the key difference between the TREC version of the task.

We consider all the documents to be of potential interest to the user. Because the query dominates each problem, the TREC task calls for deciding relevance first and novelty second – the reverse of what we will do in our system. We first identify segments that contain new information and then decide if they are interesting. In our terms, interesting is not the same as relevant, since we have no query to base relevance on.

With a query, the task is more focused, providing the system with some kind of guide for what to select, but the characteristics of the tasks vary with the kind of topic used. The sample topics suggested that deep understanding of language would help, and might even be necessary for strong performance. For example,the first sample, about the Hubble Space Telescope, asked for material about the achievements of the telescope and not material about repairs or modifications to the telescope. We know of no automated system that can classify events as achievements or not achievements in relation to an arbitrary object, here a telescope. It seemed clear that the relevance portion would dominate the task. The coordinators of the novelty track said so when the guidelines were promulgated.

Because we have no experience with relevance judgments, we chose to experiment with an unusual approach that borrowed the language analysis tools we are developing for our new information system.

1.1 New Information

NIA analyzes a document in terms of the content words and the contexts in which each one appears, and then compares documents by comparing these contexts in structure called *Concept Vectors*. In order to build these *Concept Vectors*, the system groups the words into sets of "referential equivalents" or *Concept Sets*, so that in a document about the Hubble space telescope, the words *telescope* and *instrument* would be equated and put into the same *Concept Set*. The *Concept Vectors* are creating by making lists of which *Concept Sets* co-occur with each other. These vectors are compared across documents – not sentences or clauses.

The system uses a syntactic analyzer that breaks up documents into clause-sized chunks. These are used in two different ways: 1.) potential "equivalents" are grouped together only if they appear within n clause chunks of one another, and 2.) segments of new information are identified by examining the concepts in each clause with respect to how well their corresponding vectors are covered by previously seen material. In our version of the new information task, we hypothesize that sentences are not a good unit for analysis. Rather than consider the similarity or dissimilarity of whole sentences, we are trying to efficiently decompose the documents into small chunks and discover when new relationships between entities appear.

In the TREC task, we lost that framework since the novelty part examines a collection of sentences that relate to a query, but are each individual passages taken out of context. The result is that the system we are developing is not appropriate and was ignored. In addition, we were running out of time, so that the novelty part of our task was done with a rather simple system of computing the overlap of the words sentence by sentence.

In the rest of this paper, Section 2 will discuss work related to our experiments; Section 3 will talk briefly about the system we are building; Section 4 will provide a desciption of the program used in the Novelty Track; Section 5 will review its performance; Section 7 will reflect on the lessons learned.

2 Related Work

Novelty detection is a new area of research, with roots in information retrieval, in particular first story detection under the Topic Detection and Tracking (TDT) initiative and in multi-document summarization. The task defined in the TREC Novelty Track is closer to the TDT task. Some recent work by James Allan exemplifies the extension of TDT to the passage level of documents (2001). He posit that a sentence is "useful" if it is on topic, and that a sentence is "novel" if it is not redundant with previously seen sentences. Their perspective is topic-based and the experimental corpus comes from the TDT-2 corpus, in which 60,000 news stories were assigned to some 200 news topics. After selecting 22 of these topics, annotators created lists of the events that comprised each topic and assigned each sentence to one or another event. A total of 343 events were derived from 944 articles. Two different language models for deriving "useful" information were developed, based on the probabilities that individual words of a sentence appear in on-topic sentences or articles. The models of novelty are derived in a similar way from the specific words in on-event sentences.

A numer of efforts in multi-document summarization have sought either to highlight differences or avoid redundancy. A group at CMU (Goldstein et al., 2000) uses cosine similarity of vectors in the MMR algorithm, which is cited by Allan. They seek to eliminate redundancy from their summaries with a measure similar to Allan's novelty detector. Radev attempted to create a framework for analyzing differences between sentences between sentences from different documents, with relationships such as "equivalence", "subsumption" or "contradiction" (2000).

A graph representation of several relationships between words is used to find similarities and differences between pairs of articles (Mani and Bloedorn, 1997). They recognize that sentences cannot be examined independently, without reference to other sentences in the same article. A group from Cornell and Cogentex is looking at the related problem of "discrepancy detection," in particular those of numerical differences (White et al., 2001).

The structure of the task in the Novelty Track is close to the work of Allen and that of Goldstein, although they had used a linear combination of both relevance and novelty qualities, but it requires separate computations. Our developing work views a document in a way close to Mani and Bloedorn, but unfortunately it could not be directly applied to this task.

3 Overview

The query-based structure of the Novelty Track prohibited the direct use of our system, NIA. Queries contain only general statements about the topics, and a perfectly functioning NIA would return all the details in the set of documents as *new*. But we wondered if we could apply the *Concept Sets* and the syntactic analyzer to both the relevance and novelty parts of the Novelty Track. This strategy was problematic since NIA has no machinery to determine relevance to a given query. NIA is intended to track a topic or event over time and provide updates. It assumes 1.) that the input documents are clustered appropriately, and 2.) that the user cannot predetermine what aspects of the topic or event will be interesting. But, on the other hand, the exercise might offer much insight into the performance of the tools we are developing and might ultimately be more beneficial to us than trying to quickly patch together an information retrieval system.

The borrowed tools include the lexicon used to build the *Concept Sets*. It provides what we call "potential referential equivalents", that is words that can be used to refer to one another. In addition to it, we compiled a lexicon of associated words drawn from a background corpus of news and combined these elements in a rule-based system that made a relevant/not relevant decision on each sentence in the document cluster.

3.1 Sample Sets

Like the other participants, we had only four sample sets for development, and used those to design and tune the system. The prospects were challenging. It was obvious that the four samples were quite diverse. Further, it was difficult to guess about the test data since the track organizers intended to alter the wording of some of the topics in the actual test and since we had no idea which documents might be listed as the most relevant.

We also noticed in the sample sets that the annotators' tended to favor a few of the documents. Based on that observation, we built the system to automatically decide if a few documents strongly addressed the issue in the topic. Where that was the case, we drew all our relevant sentences from those central documents.

Finally, we developed the parameters our system uses by experimenting on the sample sets. We sought to balance the recall and precision on the sample sets, and we aimed to present summaries of reasonable size, given the examples, and avoided submitting either very small or very large summaries.

The sample sets themselves were interesting. Here are some observations we made from an initial look at the problem:

Hubble Strong performance here seemed to depend on a clear idea of what is and is not an accomplishment. There were some useful key words, like data and theories, but the set contained a number of off-topic articles that were not likely to discuss Hubble's accomplish, including those on a species of squirrel and on a big earth-bound telescope being built by the Europeans.

mutual funds The system needs to know what a predictor is. There is a conflict in the language. In the *description* it says "predictors of mutual fund performance (excluding issues of costs and yields)" and in the *narrative* it says "a documnet must contain at least one factor such as: rankings, risks, yields or costs". Our initial tests were not able to suggest a strategy for this set, but it was described as atypical.

mainstreaming The interesting aspect here was that the word *mainstreaming* rarely occurred in the document set ($< 1\%$ of the sentences), but only 3 times in the relevant sentences, forcing the system to rely on the terms "children", "impairments" as well as to have an understanding of "pro" and "cons".

Mirjana Milosevic Strong performance here was attainable simply by scanning for sentences that mention the woman's first name, or nickname Mira, . Other strategies diminished these results.

4 System Features

4.1 Relevance

Most of our system-building effort went into the relevance part of the task. We settled on a rule-based approach, rather than a vector-space approach. We expected that most participants would be far more experienced in information retrieval methods and would be in a much better position to refine them to this task. Thus we viewed our submission as an opportunity to test unusual ideas that were more closely related to the thrust of our research. Admittedly, this gives our system a patchwork quality, but one that would hopeful provide valuable insight into alternative approaches.

A number of features are computed for each sentence, and sentences are selected if the rule is satisfied. We submitted five runs, using different combinations of rules and parameters. Development of this system was based almost entirely on four samples.

1. distance from a title word in a prominent role in a clause (target distance)
2. word match with a potential referential equivalent (equivalent count)
3. word match an associated word (associated count)

The first feature is binary, reflecting whether the current passage is near enough to the previous *prominent* mention in the document of a term that appears in the title. Passages were either clauses or sentences, and promiment means that the target word appears as a standalone NP before the verb.

The second and third features refer to the two lexicons mentioned above. The values are just raw counts. We observed that the clause chunks are uniformly short and that the appearence of both "equivalent" words and "associated" words is relatively rare.

We tried a number of other features, and ended up ignoring several. The ones retained were based on the various fields in the topics, such as "titles", "narratives". The three used are:

The lexicon that provides potential referential equivalents is a new version of the of a resource we have been using in NIA. There, it is used to build *Concept Sets*, which are are words linked semantically. In order to avoid the need to disambiguate among word senses, highly polysemous words are filtered out, and a distance constraint is imposed before words are grouped together in a *Concept Set*. Thus, as the text is scanned, the system checks to see if it belong to an existing set or if it will instantiate a new set. The function to accept a word for inclusion in the Relevance part is:

$$Accept(w_i) = \begin{cases} true & if \begin{array}{l} senses(w_i) < m, \\ dist(w_i, w_j) < n \end{array} \\ false & otherwise \end{cases}$$

where *senses* is the count of WordNet senses, and *dist* is the number of clauses between w_i and w_j the previous occurrence of a word in the same equivalence class.

Because the Novelty Track task required us to relate the words in a query to those in a document, we were unsure of how to modify technique, since the queries, that is the topics, are too short to allow the building of *Concept Sets*. In the end, we risked injecting noise into the decision-making and ignored the second condition for acceptance. We went forward with this strategy because it seemed to work reasonably well in tests conducted on the sample sets.

The raw equivalence lexicon is built mostly from WordNet (Miller et al., 1990), using synsets, hypernyms and hyponyms. NIA uses nouns and verbs, but we included adjectives for this effort. In the future, the lexicon will be altered with the results of corpus statistics that we are in the process of gathering. It is not clear yet whether we will keep the adjectives.

The lexicon of associated words is based on cooccurrence patterns in a background corpus. The corpus we used was from Reuters in 1996 and might have added some noise to our submission. Using the underlying TREC collections used in the track might have been more effective here, but we wanted to test using an orthogonal corpus, since in NIA we will have no knowledge of future changes in the discussion of a particular topic or event. We also used a clause-level co-occurrence standard rather than a document-level standard, since the task examines and makes decisions on short passages – sentences, which are usually composed of one, two or three clauses. We used mutual information to measure the degree of relatedness between two words.

$$MI(x,y) = log\frac{(p(x,y)}{p(x)p(y)}$$

We also added an adaptive capability to our system, given the different types of topics in the Novelty Track. These automatically assess two characteristics of the document set and adjust the system's behavior to these. In previous research in multi-document summarization, we used a similiar technique in the DEMS summarizer, Dissimilarity Engine for Multi-Document summarization (Schiffman et al., 2002), to good effect in the Document Understanding Conference 2002.

One adaptative method controls the value of the feature that check the target distance feature, the distance between the current passage and the last mention of a word in the topic title. We observed that topics did not always contain usable title words – like mainstreaming – so that the target distance would be self-defeating. For this, we measured the likelihood of finding any target word in the document set. If the total was below a threshold, we set the distance at such a large number that it no longer carried any weight.

The other adaptive method controls the number of documents that were examined. We noticed in the sample sets that in some cases a few documents dominated the selection of relevant sentences, suggesting that cluster contained some documents that were only tangentially related to the topic. In order to discern when this occurred, we used the lexicon of associated words. We computed the likelihood of finding words from any field in the topic and then computed the variance of these likelihoods across the documents in the set. If the "associated-words variance" was below a threshold, we concluded that most of them would contribute to the output of relevant sentences. Otherwise, we concluded that the document set contained some outliers that would be best to ignore.

One final strategy we adopted was to remove words that frequently appeared in a large number of topics – words like "relevant". To avoid having sentences accepted on the basis of these, we computed an inverse topic frequency value for all words in the 150 topics from which the test set would be drawn. These words were eliminated from the topics before the topics were expanded to include the referential equivalents and the associated words.

4.2 Novelty

Unfortunately the relevance part of the task took most of the time we had alloted, and with a limited time left, we adopted a very simple duplication test. From the sample topics, there was little for a novelty detector to do. We first expand each sentence by adding the referential equivalents. We did this despite the risk of eliminating novel sentences because of the appearance of unrelated senses of polysynonymous words. As we considered new sentences, we computed how well the new sentence was covered by each previous sentence and rejected those that exceeded a threshold. The mechanism we developed for NIA would have required us to reference the original documents in order to examine the context of each sentence.

5 Overfitting

We submitted all five runs that we were allowed. All used the same structure outlined in Section 4, but with different parameters. Three of them were based on clauses, that is the features were computed on the basis of the clauses recognized by our clause-tagging tool. We used these to test whether the on-line adjustments had value. The runs marked *cl35* and *cl85* in Table 1 did not try to adapt to the document set. The numbers 35 and 85 refer to the percentage of documents from which the relevant selections were drawn from. The documents are ordered according to their distributions of associated words. The run marked *clfx* automatically selected either the .35 or .85 figure according the variance of the likelihood of finding an associated word in the documents.

The *sent* run computed the features over sentences, and the *merg* concatenated the sentence following any sentences that scored high, to test the possibility that segmenting documents might be a valuable idea. Both of these used the automative adaptation mechanism.

It appears that we were lulled by a painful instance of overfitting. The devlopment of our system was closely guided by its performance on three of the sample sets. The first, topic 303, was described as typical of the entire test. Table 2 shows how the system performed on

	Relevant			New		
	P	R	P*R	P	R	P*R
cl35	.07	.04	.006	.07	.04	.005
cl85	.09	.07	.009	.08	.05	.007
clfx	.07	.12	.012	.07	.09	.009
sent	.11	.09	.012	.12	.09	.012
merg	.11	.15	.020	.09	.10	.013
humans			.191			.170
random			.006			.004
best system			$< .095$			< 0.85

Table 1: Precision and Recall of our five runs, humans and random

the relevance part. The results seemed to be sufficient on this difficult problem. We didn't think we had a top system, but were satisfied with what we saw.

We didn't include topic 359 for several reasons. In our early experiments, it seemed to be impossible to match any of the human selection and futher more it contained a contradiction: Description 2 wants to exclude costs and yields, but the Narrative wants to include them.

Our results were disappointing even though we did not expect much at the outset. The organizers of the Novelty Track reported that human annotators tested against each other had achieved a score of 0.19 – this is the product of the standard measures of precision and recall – $Score = Prec * Recall$. They said that the best submission was less than half that of the humans, but Table 1 clearly shows all of our runs were far below that.

If an oracle program were able to choose the best system for each topic, the combined score averaged together would be 0.134 on the relevant part and 0.120 on the novelty part. Since this score was a good deal better than the best system, no one system was consistently at the very top.

The topic sets also varied widely, and some were difficult for all systems, other much easier. Averaging the scores by all systems for each topics showed a wide range, indicating some sets were managable for a number of systems, while others were nearly impossible for all of them. Assuming that the average of all systems indicates the degree of difficult we have:

On average, the novelty task proved to be much harder. There was a striking drop off in the average scores. This is surprising since the annotators eliminated very few relevant sentences in creating their list of new sentences. In fact, a baseline that does nothing – that does not eliminate any relevant sentence – would have a precision of .91 and a recall of .99. (The recall appears to be short of 1.00 because the relevant and new lists were swapped in two cases.)

Just before the paper submission deadline, the Novelty Track organizers restated the results, using the standard F-measure instead of the product of precision and recall.

$$F = \frac{2PR}{P + R}$$

The recalculation raised the single-value scores of all groups, and squashed the results into a much narrower range for the automatic systems, as Table 5 shows. The recalcuation also tended to eliminate the size of the advantage to systems that generated larger summaries.

6 Recent Experiments

To explore performance in the novelty part further, we counted duplicates found, rather than novel sentences found. In the formal task, the scores in the first part address the question of "How many of the relevant sentences can the system find?" The scores for the second part address the similar question of "How many relevant (and nonduplicative) sentences can the system find?" The question makes more senses in the relevance part where only a small portion of the sentences are judged relevant. In the four sample top-

Results on Sample Set									
	Clause-based *clfx* run			Single sentence *sent* run			Paired sentences *merg* run		
Topic	Prec	Recall	P*R	Prec	Recall	P*R	Prec	Recall	P*R
303	.636	.438	.279	.667	.250	.167	.261	.375	.098
379	.194	.235	.046	.250	.216	.054	.238	.294	.070
423	.211	.160	.034	.786	.147	.116	.545	.320	.174
	average $p * r = .120$			average $p * r = .112$			average $p * r = .114$		

Table 2: Precision and Recall achieved by our system across the three sample topics for detection of relevance.

Relevance			
Easiest Topics		Hardest Topics	
Topic	Score	Topic	Score
368	0.262	312	0.019
397	0.247	381	0.018
394	0.193	305	0.018
365	0.189	432	0.017
369	0.167	420	0.016

Table 3: Average $P * R$ scores on the Relevance part show a wide range of difficulty.

Novelty			
Easiest Topics		Hardest Topics	
368	0.127	445	0.005
397	0.108	312	0.005
394	0.103	432	0.003
365	0.089	377	0.002
364	0.080	420	0.0002

Table 4: Five highest average $P * R$ scores in the Novelty Part of the task.

F-Measures	
Summary generator	F-measure relevant
Humans	0.371
Top Sys	0.235
Best Novcol	0.126
Random	0.040

Table 5: Restatement of some results on the Relevant part of the task in terms of the standard F-measure. The 10 top scores plus human and random results were distributed by NIST before the paper-submissions were due. The Novcol were recomputed.

ics given to participants, about 6% of the sentences were accepted as relevant. The situation was the reverse for the novelty side, where nearly all the relevant sentences were considered novel. In the test, the annotators removed only 106 of 1,347 sentences were removed as duplicative. With a lopsided test set, it is hard to beat the baseline of "do nothing." So we recast the question into "How many duplicative sentences can the system find?" We then computed precision and recall for a number of baselines, including a bag of words approach, TF*IDF, longest common subsequence, and Simfinder, another tool for measuring similarity between sentences developed at Columbia(Hatzivassiloglou et al., 2001). Table 6 shows that our semantic module outperformed the other methods. We show the results when the parameters for the various methods made a reasonable number of selections. By making the duplication thresholds low enough, most of these methods will choose a large proportion of sentences as duplicates and achieve a high recall.

Here are descriptions of the methods presented in Table 6..

novcol Our semantic module applied to whole sentences.

sequent The longest common subsequence of words, as a percentage of sentence length.

wordbag A unweighted bag of words approach, using overlap.

similar The Simfinder utility..

tfidf TF*IDF metric [1] and computing cosine similarity. Document frequency values were taken from the set of articles for the topics.

random An extrapolation of random results by computed the expected value of 106 selections.

7 Conclusion and Future Work

One positive lesson learned in this exercise is that the adaptive strategy appears to have considerable value. Without sufficient training data, it was impossible to explore and sharpen the technique, yet it clearly improved our results in the runs where it was applied, despite having only a rough idea of the parameters to use. In addition the range of averages across the topics suggests that a one-size-fits-all approach is not the best.

Our experiments after the evaluation show there is a value using semantic information in detecting similarity and dissimilarity. This was not so clear about our application in the relevance part of semantic data – in the form of the lexicon of referential equivalents. We were hampered because our system was unable to apply the lexicon in the way it is used in our NIA system, where the expansion of the sets is limited by the context in the documents to be summarized. Since the topics were too small to provide any context, the lexicon was used without distance constraints. But in the more straightforward task of detecting duplication, the semantic information without those constraints

An assessment of using associated words – those obtained by co-occurrence studies – was clouded by the fact that the data was drawn from a much different collection of background documents. This was due to a lack of time. We had a collection of Reuters news wire already parsed, and would have had to delay experimentation if we had waited to parse the TREC collections used in the Novelty Track. We are planning to create a new lexicon based on the TREC documents to compare against the results here.

The Novelty Track also confirmed how difficult the task is. The subjectivity of the annotation greatly complicates the conclusions that can be drawn. Judging from the cross annotator scores, inter-annotator agreement was quite low, and the choice of annotator may have had a large effect on the results in various sets.

We were also struck by the fact that many, but certainly not all, topics included some instruction about was not relevant. We blocked those that were found in such negative sentences from being expanded if they were not already found in the positive sentences – however these were few in number in the sample sets. We did test a feature of noting the presence of negative terms in the passages, but where it did affect the outcome, it was detrimental as often as helpful. Yet, we think the idea of trying to categorize the queries, that is the topics, is worth further experimentation.

[1] $weight = (1 + log(tf))log(idf)$

	Matched	Sys Tries	Hum Picks	Prec	Recall	P*R
novcol	34	139	106	0.2446	0.3208	0.0785
sequent	30	156	106	0.1923	0.2830	0.0544
wordbag	24	107	106	0.2243	0.2264	0.0508
similar	25	158	106	0.1582	0.2348	0.0373
tfidf	14	126	106	0.1111	0.1321	0.0147
random	8.3	106	106	0.0783	0.0783	0.0061
do-nothing	0	0	106	0	0	0

Table 6: A comparison of results on duplication detection between the semantic module and several word-based methods and a system that would choose at random. Note "do-nothing" gets zero because it selects no duplicates to reject.

References

James Allan, Rahul Gupta, and Vikas Khandelwal. 2001. Temporal summaries of news topics. In *Proceedings of the ACM-SIGIR Conference.*

Jade Goldstein, Vibhu Mittal, Jaime Carbonell, and Mark Kantrowitz. 2000. Multi-document summarization by sentence extraction. In *Proceedings of ANLP/NAACL-2000 Workshop on Automatic Summarization.*

Vasileios Hatzivassiloglou, Judith L. Klavans, Melissa L. Holcombe, Regina Barzilay, Min-Yen Kan, and Kathleen R. McKeown. 2001. Simfinder: A flexible clustering tool for summarization. In *Proceedings of the NAACL 2001 Workshop on Automatic Summarization.*

Inderjeet Mani and Eric Bloedorn. 1997. Multi-document summarization by graph search and matching. In *Proceedings, American Association for Artificial Intelligence 1997.*

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1990. Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography (special issue)*, 3(4):235–312.

Dragomir Radev. 2000. A common theory of information fusion from multiple text sources, step one: Cross-document structure. In *1st ACL SIGDIAL Workshop on Discourse and Dialogue.*

Barry Schiffman, Ani Nenkova, and Kathleen McKeown. 2002. Experiments in multidocument summarization. In *Proceedings of the Human Language Technology Conference.*

Michael White, Claire Cardie, Vincent Ng, Krii Wagstaff, and Daryl McCullough. 2001. Detecting discrepancies and improving intelligibility: Two preliminary evaluations of riptides. In *Proceedings of the Document Understanding Conference (DUC01).*

TREC11 Web and Interactive Tracks at CSIRO

Nick Craswell[1] *David Hawking*[1] *James Thom*[2]
Trystan Upstill[3] *Ross Wilkinson*[4] *Mingfang Wu*[4]

[1]Enterprise Search
[4]Technologies for Electronic Documents
CSIRO Mathematical and Information Sciences
{Nick.Craswell; David.Hawking; Ross.Wilkinson; Mingfang.Wu}@csiro.au

[2]School of Computer Science and Information Technology, RMIT University
jat@cs.rmit.edu.au

[3]Department of Computer Science, Australian National University
Trystan.Upstill@anu.edu.au

1. Overview

This year, the CSIRO teams participated and completed runs in two tracks: web and interactive.

Our web track participation was a preliminary exploration of forms of evidence which might be useful for named page finding and topic distillation. For this reason, we made heavy use of evidence other than page content in our runs.

In the interactive track, we continue to focus on answer organization issues, aiming to investigate the usefulness of the knowledge about "organizational structure" in organizing and delivering the retrieved documents. For the collection of the US government (.gov domain) web documents, we used their level two domain labels and their corresponding organization names to categorize the retrieved documents. For example, documents from the "nih.gov" domain will be put into the "National Institutes of Health (nih)" category. We compared this delivery method with the traditional ranked list. The preliminary results indicate that subjects achieved a significantly better performance with the category interface at the end of fifteen minutes search, however, there is no significant difference between the two methods during the first five or ten minutes. The experiment result also shows that the category interface assisted subjects answer the more complex topics as time increases.

2. The web track

2.1. Topic distillation

In topic distillation we used the following forms of evidence:

- BM25 on content. Pages returned should be relevant. We indexed the .GOV corpus and applied BM25, sometimes with stemming sometimes without.
- BM25 on content and referring anchor text. An alternative to content-only BM25 is to include referring anchor text words in the BM25 calculation (content and anchors).

- In-link counting and filtering. We expected pages with more in-links to be potentially better answers, and we differentiated between on-host and off-host links. We also eliminated many results on the grounds that they had insufficient in-links.

- URL length. We expected short URLs to be better answers than long URLs

- BM25 score aggregation. We expected sites with many BM25-matching pages to be better than those with few.

Table 1 reports the results for our topic distillation runs. Our (non-submitted) content-only achieved better performance than any of the submitted runs that included "distillation evidence".

Table 1 Runs for topic distillation

Run	*P@10*	*BM25 content only*	*BM25 content and anchors*	*In-link counting and filtering*	*URL length*	*BM25 aggregation*
csiro02td1	0.1000	y		y	y	
csiro02td2	0.0714		y	y		
csiro02td3	0.0184	y		y	y	y
csiro02td4	0.0184		y	y		y
csiro02td5	0.0939	y (stem)		y	y	
csiro02unoff	0.1959	y				

In this year's topic distillation task, the focus on local page content relevance ("BM25 content only") was probably too high for our non-content and aggregation methods to succeed (our "distillation evidence"). We expected most correct answers to be shallow URLs of sites containing much useful content. In fact, correct answers were deeper, and our aggregation method for finding sites rich with relevant information was actually quite harmful (runs 3 and 4). The focus on page content is borne out by the improvement in effectiveness achieved when we apply simple BM25 in an unofficial run (csiro02unoff). To perform better in this year's task, we should have put less (or no) emphasis on distillation evidence and far more emphasis on relevance. However, we also believe that in some Web search situations, the distillation evidence would be more important than it was in this year's task.

2.2. Named Page Finding

In our named page finding experiments we used the following forms of evidence:

- BM25 on content and/or anchor text. We indexed the .GOV corpus and applied BM25 to document content and to surrogate documents that contained all anchor text pointing to a page. Stemming of query terms was also employed.

- Extra Title Weighting. To bias our results towards what we thought would be page naming text we put further emphasis on document titles.

- PageRank. To see whether link recommendation could be used to improve results we incorporated this link popularity measure [3].

Table 2 shows the results for the named page finding runs. The BM25 content-only submission performed the best. We tried combining content evidence with anchor-text and PageRank but both combinations harmed retrieval effectiveness.

Table 2 Runs for named page finding

Run	*ARR*	*S@10*	*BM25*	*Stemming*	*Extra Title Weighting*	*Small Crawl PageRank*
csiro02np01	0.573	0.77	Content			
csiro02np02	0.241	0.34	Anchor text			
csiro02np03	0.416	0.59	Content		y	
csiro02np04	0.318	0.51	Content and anchor text	y	y	
csiro02np16	0.307	0.49	Content and anchor text	y	y	y

Prior to submission we generated 20 training queries and found content with extra title weighting performed best. We expected page titles to be important evidence in named page finding, however this appeared not to be the case – in fact extra title weighting for the TREC queries appeared to reduce effectiveness (run 1 vs run 3). While there was some anchor text evidence present for the query set (run 2) when we combined this evidence with content (runs 4 and 16) results were noticeably worse than for the content-only run (run 1). PageRank harmed retrieval effectiveness (run 16 vs run 4).

3. The interactive track

On the Internet, the information source and its information provider indicate not only the quality and credibility of the information, but also the type and content of the information. When people try to access information from an organization's website, they very often try to match their mental model about that organization with their information needs. They can usually identify a few related departments in that organization, and search the information within these departments.

We can consider the whole worldwide web as the web site of a global organization with a hierarchical structure. Documents in this space could be categorized by their "functional departments" corresponding to their domain names. For example, the level one domain labels can categorize the documents into government (.gov), university (.edu), military (.mil), and commercial (.com) etc. (In fact, they should be the level two domain labels, with the level one label of .us) ; the level two domain label can be used to further categorize the documents within the first level domain.

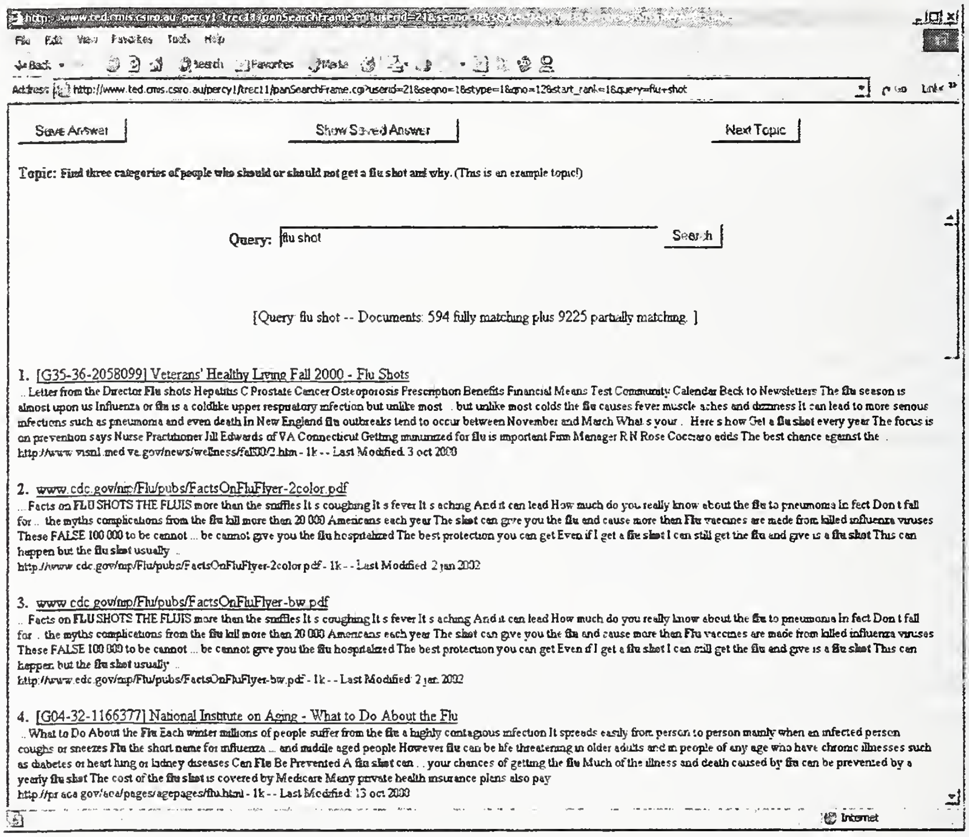
In this year's interactive track, all documents in the collection are gathered from the US government domain (.gov). The test topics also cover various areas, such as government policy, medicine/health and travel. To this collection and the topic set, our intuition was to organize and delivery the retrieved documents according to the US government functional (or departmental) structure. We intent to use this dynamically generated organizational structure to organize the distributed documents retrieved from the web, and guide users to focus their attentions on the information sources and/or information providers. We hypothesized that this structure (called categorization structure) would serve as a better guide for a user to locate relevant and authoritative information than the traditional ranked list, thus improving the user's performance with the search tasks.

3.1. Experimental setting

3.1.1. Delivery interfaces

The Panoptic [1,2] is used as the back-end search engine in both delivery methods. In the categorization delivery method, the categoriser classifies the retrieved documents according to the level two domain labels. Each category label is obtained by expanding the domain label into its owner's organizational name through the "whois" server (http://www.whois.nic.gov). For example, all documents from the "nih.gov" domain will be put into the "National Institutes of Health" category. The documents in a category are ranked according to their original rank in the returned ranked list, and the categories are ranked according to the original rank of the first document of each category. The category interface shows the first category by default.

The interfaces for the two different delivery methods are shown in Figure 1 and Figure 2. We have been trying to keep the two interfaces as consistent as possible, differing only in their presentation of the alternate structures. Both interfaces are divided into three areas: the top area shows the current search topic and provides three buttons for the subjects to save answer and move on to the next topic. The middle area is the query area that has a query box and information on query word matching. The bottom area is the main area that shows either the ranked list or the categorized result.


File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.ted.cmis.csiro.au/percy1/trec11/panSearchFrame.cgi?userid=21&seqno=1&stype=1&qno=12&start_rank=1&query=flu+shot
Go Links
Save Answer
Show Saved Answer
Next Topic
Search
Internet

Topic: Find three categories of people who should or should not get a flu shot and why. (This is an example topic!)

Query: flu shot

[Query flu shot -- Documents: 594 fully matching plus 9225 partially matching.]

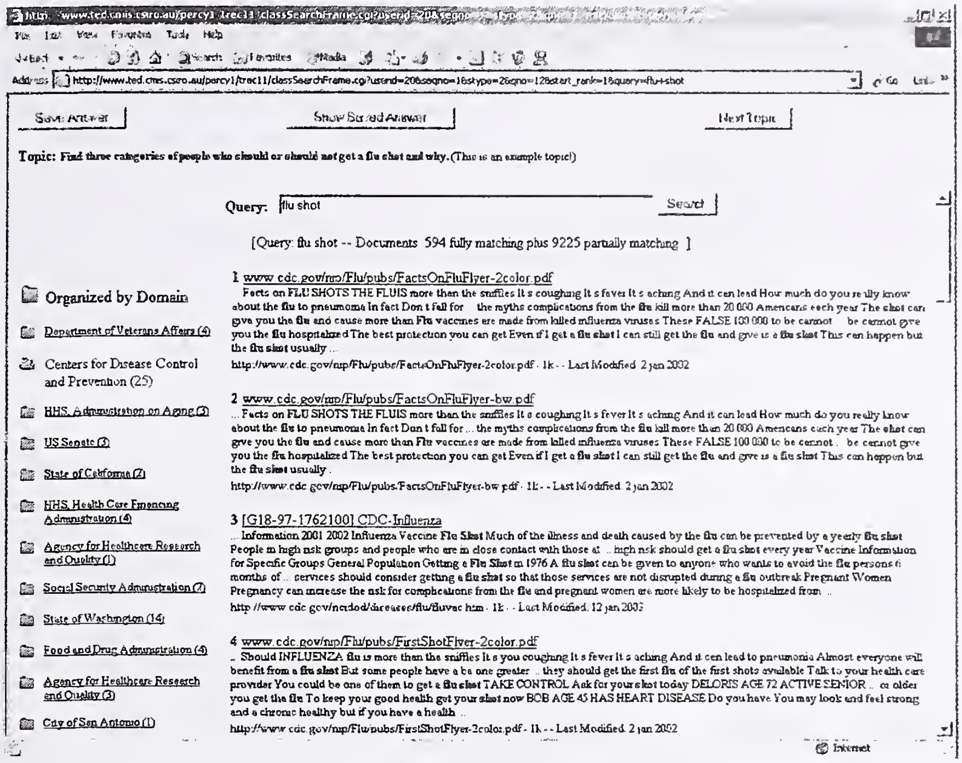
1. [G35-36-2058099] Veterans' Healthy Living Fall 2000 - Flu Shots
.. Letter from the Director Flu shots Hepatitis C Prostate Cancer Osteoporosis Prescription Benefits Financial Means Test Community Calendar Back to Newsletters The flu season is almost upon us Influenza or flu is a coldlike upper respiratory infection but unlike most .. but unlike most colds the flu causes fever muscle aches and dizziness It can lead to more serious infections such as pneumonia and even death In New England flu outbreaks tend to occur between November and March What s your .. Here s how Get a flu shot every year The focus is on prevention says Nurse Practitioner Jill Edwards of VA Connecticut Getting immunized for flu is important Firm Manager R N Rose Coccaro adds The best chance against the ..
http://www.visn1.med.va.gov/news/wellness/fall00/2.htm - 1k -- Last Modified 3 oct 2000

2. www.cdc.gov/nip/Flu/pubs/FactsOnFluFlyer-2color.pdf
... Facts on FLU SHOTS THE FLU IS more than the sniffles It s coughing It s fever It s aching And it can lead How much do you really know about the flu to pneumonia In fact Don t fall for .. the myths complications from the flu kill more than 20 000 Americans each year The shot can give you the flu and cause more than Flu vaccines are made from killed influenza viruses These FALSE 100 000 to be cannot ... be cannot give you the flu hospitalized The best protection you can get Even if I get a flu shot I can still get the flu and give is a flu shot This can happen but the flu shot usually ..
http://www.cdc.gov/nip/Flu/pubs/FactsOnFluFlyer-2color.pdf - 1k -- Last Modified 2 jan 2002

3. www.cdc.gov/nip/Flu/pubs/FactsOnFluFlyer-bw.pdf
.. Facts on FLU SHOTS THE FLU IS more than the sniffles It s coughing It s fever It s aching And it can lead How much do you really know about the flu to pneumonia In fact Don t fall for .. the myths complications from the flu kill more than 20 000 Americans each year The shot can give you the flu and cause more than Flu vaccines are made from killed influenza viruses These FALSE 100 000 to be cannot ... be cannot give you the flu hospitalized The best protection you can get Even if I get a flu shot I can still get the flu and give is a flu shot This can happen but the flu shot usually ..
http://www.cdc.gov/nip/Flu/pubs/FactsOnFluFlyer-bw.pdf - 1k -- Last Modified 2 jan 2002

4. [G04-32-1166377] National Institute on Aging - What to Do About the Flu
.. What to Do About the Flu Each winter millions of people suffer from the flu a highly contagious infection It spreads easily from person to person mainly when an infected person coughs or sneezes Flu the short name for influenza ... and middle aged people However flu can be life threatening in older adults and in people of any age who have chronic illnesses such as diabetes or heart lung or kidney diseases Can Flu Be Prevented A flu shot can .. your chances of getting the flu Much of the illness and death caused by flu can be prevented by a yearly flu shot The cost of the flu shot is covered by Medicare Many private health insurance plans also pay
http://pr.aoa.gov/aoa/pages/agepages/flu.html - 1k -- Last Modified 13 oct 2000

Figure 1 The delivery interface for the ranked list


http://www.ted.cmis.csiro.au/percy1/trec11/classSearchFrame.cgi?userid=20&seqno
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.ted.cmis.csiro.au/percy1/trec11/classSearchFrame.cgi?userid=20&seqno=1&stype=2&qno=12&start_rank=1&query=flu+shot
Go Links
Save Answer
Show Saved Answer
Next Topic
Search
Internet

Topic: **Find three categories of people who should or should not get a flu shot and why.** (This is an example topic!)

Query: flu shot

[Query: flu shot -- Documents 594 fully matching plus 9225 partially matching]

Organized by Domain

- Department of Veterans Affairs (4)
- Centers for Disease Control and Prevention (25)
- HHS, Administration on Aging (3)
- US Senate (3)
- State of California (2)
- HHS, Health Care Financing Administration (4)
- Agency for Healthcare Research and Quality (1)
- Social Security Administration (2)
- State of Washington (14)
- Food and Drug Administration (4)
- Agency for Healthcare Research and Quality (3)
- City of San Antonio (1)

1 www.cdc.gov/nip/Flu/pubs/FactsOnFluFlyer-2color.pdf
Facts on FLU SHOTS THE FLUIS more then the sniffles It s coughing It s fever It s aching And it can lead How much do you really know about the flu to pneumonia In fact Don t fall for the myths complications from the flu kill more than 20 000 Americans each year The shot can give you the flu and cause more than Flu vaccines are made from killed influenza viruses These FALSE 100 000 to be cannot be cannot give you the flu hospitalized The best protection you can get Even if I get a flu shot I can still get the flu and give is a flu shot This can happen but the flu shot usually ...
http://www.cdc.gov/nip/Flu/pubs/FactsOnFluFlyer-2color.pdf - 1k - - Last Modified 2 jan 2002

2 www.cdc.gov/nip/Flu/pubs/FactsOnFluFlyer-bw.pdf
... Facts on FLU SHOTS THE FLUIS more than the sniffles It s coughing It s fever It s aching And it can lead How much do you really know about the flu to pneumonia In fact Don t fall for ... the myths complications from the flu kill more than 20 000 Americans each year The shot can give you the flu and cause more than Flu vaccines are made from killed influenza viruses These FALSE 100 000 to be cannot . be cannot give you the flu hospitalized The best protection you can get Even if I get a flu shot I can still get the flu and give is a flu shot This can happen but the flu shot usually .
http://www.cdc.gov/nip/Flu/pubs/FactsOnFluFlyer-bw.pdf - 1k - - Last Modified 2 jan 2002

3 [G18-97-1762100] CDC-Influenza
... Information 2001 2002 Influenza Vaccine Flu Shot Much of the illness and death caused by the flu can be prevented by a yearly flu shot People in high risk groups and people who are in close contact with those at .. high risk should get a flu shot every year Vaccine Information for Specific Groups General Population Getting a Flu Shot in 1976 A flu shot can be given to anyone who wants to avoid the flu persons 6 months of .. services should consider getting a flu shot so that those services are not disrupted during a flu outbreak Pregnant Women Pregnancy can increase the risk for complications from the flu and pregnant women are more likely to be hospitalized from ..
http://www.cdc.gov/ncidod/diseases/flu/fluvac.htm - 1k - - Last Modified 12 jan 2002

4 www.cdc.gov/nip/Flu/pubs/FirstShotFlyer-2color.pdf
.. Should INFLUENZA flu is more than the sniffles It s you coughing It s fever It s aching And it can lead to pneumonia Almost everyone will benefit from a flu shot But some people have a be one greater .. they should get the first flu of the first shots available Talk to your health care provider You could be one of them to get a flu shot TAKE CONTROL Ask for your shot today DELORIS AGE 72 ACTIVE SENIOR .. or older you get the flu To keep your good health get your shot now BOB AGE 45 HAS HEART DISEASE Do you have You may look and feel strong and a chronic healthy but if you have a health ..
http://www.cdc.gov/nip/Flu/pubs/FirstShotFlyer-2color.pdf - 1k - - Last Modified 2 jan 2002

Figure 2 The delivery interface for domain categorization

3.1.2. Experimental procedures

During the experiment, all subjects are asked to follow the following procedures.

- Subjects filled in the pre-search questionnaire about their demographic information and their search experience.
- Subjects were then shown the two experimental interfaces, and were free to ask any question related to the use of the two interfaces.
- Subjects were assigned to the experimental design that was used by all participant groups in the interactive track. In this experimental design, subjects searched four topics on each interface, the sequence of interface and topics varied among subjects. A complete such a design requires a group of 16 subjects.
- Prior to each interface, subjects had hands-on practice with an example topic, and got familiar with each interface.
- Prior to each interface, the query "information retrieval" was issued by the corresponding system automatically to calibrate the difference between two systems' response time. Subjects were asked to click the "Next Topic" button when they saw the search result appeared. The average response times are 6.8 seconds for the ranked list interface and 8.3 seconds for the category interface.
- Prior to the search of each topic, subjects were required to fill in a pre-search questionnaire about their familiarity with the topic. After the search of the topic, subjects filled in a post-search questionnaire about their experience of that particular search topic.
- Subjects filled in a post-system questionnaire after each interface.

- Subjects filled in an exit questionnaire in the end of the experiment.

At any time during a topic search, subjects could move on to the next topic whenever they found the required answer and were satisfied with what they have found. We encouraged our subjects to find answers to a topic within ten minutes, however they could have an extra five minutes in case they could not find the required answer in the first ten minutes and want to continue their search.

Transaction logging, questionnaire, and screen recording are the main methods to collect data. During each search session, every significant event - such as document read, the instance saved and the supporting source document and the query sent - was automatically captured. Questionnaires are those common to all participant groups in the interactive track. Screen recording was used to capture the search process for further detailed analysis.

3.1.3. Subjects

All our sixteen subjects were university students. These subjects came from various backgrounds, such as computer science, media study, law and mechanical engineering. Of the sixteen subjects, fourteen are male and two are female. Fifteen of them are in the age group 18-27 years, only one is in the age group 38-47 years. Table 3 lists subjects' responses to the selected questions from the pre-search question (all are on 7-point Likert scale). From the table, we can see that our subjects search the web very often (Q1, mean=5.81), can usually find what they are looking for (Q5, mean=5.38), and generally regard themselves as experienced searcher (Q10, mean=4.73). Comparatively, subjects use the search box (Q6, mean=5.19) more often than browsing mechanism (Q7, mean=4.06). These subjects very often search for information related to assignments (Q8-1, mean=5.38) and entertainment (Q8-6, mean=5.19), while search less on shopping (Q8-2, mean=3.19), government policy (Q8-5, mean=3.06), and traveling (Q8-3, mean=2.94), and least on medical/health (Q8-4, mean=1.94). (While our test topics cover the government policy, traveling, and medical/health.)

Table 3 The selected questions[1] from the pre-search questionnaire.

	Q1	Q5	Q6	Q7	Q8-1	Q8-2	Q8-3	Q8-4	Q8-5	Q8-6	Q10	Q11
Mean	5.81	5.38	5.19	4.06	5.38	3.19	2.94	1.94	3.06	5.19	4.73	4.53
Std	1.05	0.96	1.83	1.98	1.09	1.64	1.06	1.18	1.65	1.42	1.34	1.61

3.2. Results

3.2.1. Performance with two interfaces

The effectiveness of the two interfaces is measured by the success rate: the ratio of the correctly saved instances. There are two types of topics in this year's interactive track. Type I topic is "find X instance of …". Type II topic is "find a website that is a good resource on Y". For the topics of type I (topic 1, 2, 4, 5 and 6), each instance correctly identified and supported by a document will be given a score 1/n, where n is the number of required instances for the search topic. Type II topic can be regarded as a special case of the type I where the required instance is 1. So for the topics of type II (topic 3, 7 and 8), the score is binary: 1 - for the correctly identified website, and 0 – for a website that does not give information on the topic. (A 5- or 7- point Likert scale could be used to judge the degree of "goodness" of the saved website. However, this kind of judgement might be too subjective to reach consistence. So we adopted the binary score.)

Table 4 shows the subjects' search performance at three period cut-off - after five minutes, ten minutes, and fifteen minutes. On the average, the performance with the category interface is

[1] Questions are listed in the Appendix I. All responses are on a 7-point Likert scale.

Table 4 Subjects' search performance per topic at three period cut-off

Topics		1	2	3	4	5	6	7	8	Mean	Std	p <
5Min	List	0.38	0.08	0.75	0.44	0.42	0.21	0.13	0.63	0.38	0.24	0.26
	Cate	0.38	0.04	0.63	0.22	0.25	0.42	0.00	0.63	0.32	0.24	
10Min	List	0.38	0.25	1.00	0.75	0.79	0.42	0.38	0.88	0.61	0.28	0.48
	Cate	0.58	0.33	1.00	0.53	1.00	0.63	0.25	0.88	0.65	0.29	
15Min	List	0.38	0.54	1.00	0.84	0.92	0.50	0.50	0.88	0.69	0.24	0.05
	Cate	0.75	0.63	1.00	0.75	1.00	0.88	0.63	1.00	0.83	0.16	

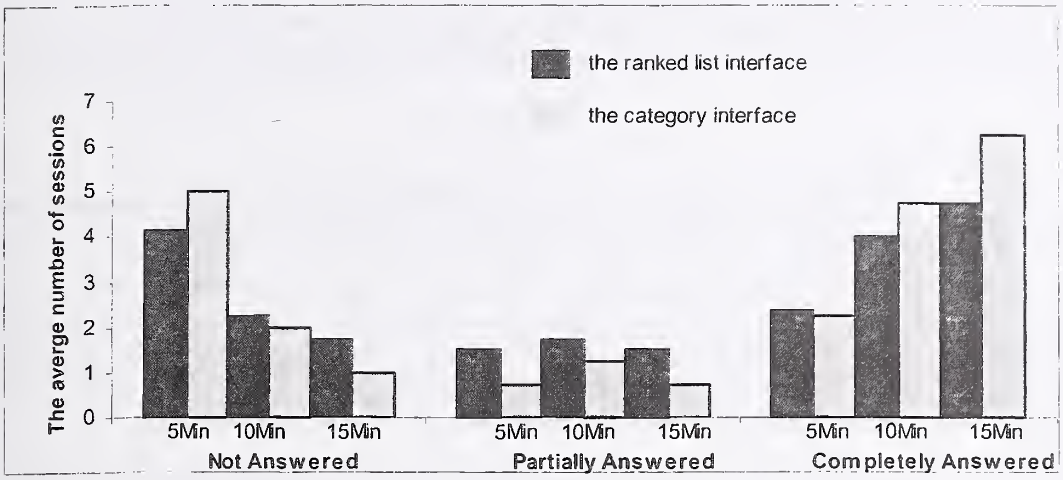


Figure 3 The completeness of the saved answers

lower than that with the ranked list interface at the end of the five minutes, higher than the ranked list interface at the end of the ten minutes, and significantly outperform the ranked list interface at the end of fifteen minutes. (two tailed, paired t-text)

Before the search of each topic, subjects were asked about their level of familiarity with the topic on a 7-point Likert scale. On the average, our subjects have low familiarity with all topics (Ranked list: Mean = 2.14, Std = 0.70; Category: Mean = 2.19, Std = 0.72). Although the correlation between the success rate with the ranked list and the familiarity ($r = 0.51$) is higher than the correlation between the success rate with category and the familiarity ($r = -0.0004$), nevertheless, neither of the correlations is significant.

Figure 3 shows the breakdown of the success rate according to the sessions in which a question is either "not answered", "partially answered", or "completely answered" at three cut-off periods.

At the end of the first five minutes, the number of "not answered" sessions with the category interface is more than the number of those with the ranked list interface. However, with the increase in time spent, the number of "not answered" sessions with the category interface decreases, although the difference is not significant at each cut-off period.

At the end of each cut-off period, the number of "partially answered" session with the category interface is always less than the number of those with the ranked list interface, although the difference is not significant either.

At the first cut-off period, subjects have less "completely answered" sessions with the category interface than that with the ranked list interface (not significant). However, at the second and third cut-off period, subjects have significantly more "completely answered" sessions using the category interface ($p < 0.05$ at tenth minute, and $p < 0.01$ at the fifteenth minute). Looking at topic by topic at the fifteenth minute, the category interface is performing better for 7 out of 8 topics. In the only exceptional topic – the topic 3, the two interfaces performed the same with the same number of "completely answered" sessions). For the topics 1, 2, and 6, the number of "completely answered" sessions with the category interface is twice that with ranked list interface. Here the topic 1, 2 and 6 are all of the type I.

We had assumed that the type I topics might need to gather instances from multiple documents, but this is not always the case – sometimes a document may contain enough information to cover all required instances. Table 5 shows the distribution of the "completely answered" sessions from either the multiple documents or one document only. In four out of five such type I topics, there are more sessions with the category interface in which the saved answers come from multiple documents. This may suggest that the category interface is more helpful for the more complicated tasks.

3.2.2. *Subject's effort*

The subject's effort for getting an answer is measured by the time, the number of documents read, and the number of queries sent in order to get a complete answer or reach the end of each session.

Table 6 shows the average time spent in order to get a complete answer by the two quickest subjects using each interfaces. On the average, the quickest two subjects using the category interface took less time than the quickest two using the list interface, but the difference is not significant here. If we look topic by topic, the two subjects are quicker using the category interface only for three topics – the topic 1, 2 and 4; there three topics are of type I.

Table 7 shows the interaction between the subject and the interface. On the average, subjects read more documents with the category interface (Mean=4.81) than with the ranked list interface (Mean=4.74), but sent less queries with the category interface (Mean=3.0) than with the ranked list interface (Mean=3.54). This indicate that the ranked list interface may encourage subjects to rephrase queries, while the category interface may encourage subjects to browse the answer structure, thus read more documents.

Usually subjects read and saved documents high in rank from the ranked list interface - the average rank of the read and saved documents is 4.97 and 4.87 respectively. While the average rank of the read and saved documents from the category interface is 19.10 and 16.5 respectively. This may indicate that the category interface may be able to bring relevant (or related) documents in a category; these documents may scatter in the ranked list, while a subject may not go that far to get that relevant document with the ranked list interface.

Table 5 The source of the complete answers

(M: from multiple documents; S: from one document only)

Topic	1		2		4		5		6	
	M	S	M	S	M	S	M	S	M	S
List	0	3	0	2	2	2	1	6	2	1
Cate	1	5	2	2	2	3	3	5	5	1

Table 6 The average time (in minute) spent to get a complete answer by the two quickest subjects.

Topic	1	2	3	4	5	6	7	8	Mean
List	4.03	12.85	0.75	6.59	2.01	6.08	4.55	1.32	4.77
Cate	3.54	9.03	2.68	4.36	3.93	6.24	5.82	1.44	4.61

Table 7 Subject-interface interaction

		Mean	Std	P < (2 tail t-test)
Number of documents read	List Category	4.74 4.81	2.52 1.54	NS
Number of queries	List Category	3.54 3.00	1.93 1.39	NS
The ranking of the read documents	List Category	4.97 19.10	2.05 7.94	0.0007
The ranking of the saved documents	List Category	4.87 16.5	2.62 14.32	0.04

Table 8 Subjects' response to the post-search questionnaire

	PS1 (easy to start)	PS2 (east to search)	PS3 (satisfaction)	PS4 (timeliness)	PS5 (knowledge helped?)	PS6 (learn something new)
List	4.59	4.24	4.49	5.22	2.22	4.13
Cate	4.11	3.97	4.25	4.19	2.19	3.69

3.2.3. *Subject's satisfaction*

After the search of each topic, subjects filled in a post-search questionnaire that was to get the subject's satisfaction of that particular search topic. Table 8 shows the subjects' response to each question. For all questions, the average response from the subjects using category interface is lower than that from the subjects using the ranked list interface, although no significant difference is found between the two interfaces for any questions.

We checked the correlation between the each question and the success rate, significant positive correlation is found only between the PS3 (satisfaction) and the success rate (in both interfaces, $r = 0.69$, significance at 0.05). That may be truism: if subjects saved more answers, they are getting more satisfied.

In the exit questionnaire, when the subjects were asked about which of the systems they like the best overall, 11 subjects chose the category interface, 3 subjects chose ranked list interface, while the remaining 2 thinking there is no difference between the two interfaces.

3.3. Discussion

Our experimental results indicate that the users may be able to find the answer quicker with the ranked list interface for those easy search tasks where the search engine is able to bring the relevant documents on the top of the ranked list. However, for more complicated tasks where an answer is to be synthesized from multiple documents, and those documents are scattered along the ranked list, the user may perform better with the category interface. This performance is achieved by spending longer reading or browsing time. One possible reason might be related to the categorization structure itself: the current one-level flat structure may not be very clear to the subjects. It could be enhanced by having a multi-level hierarchical structure closely reflecting the US governmental structure.

4. References

[1] CSIRO, http://www.panopticsearch.com

[2] David Hawking, Peter Bailey and Nick Craswell. Efficient and Flexible Search Using Text and Metadata. CSIRO Mathematical and Information Sciences, TR2000-83, http://www.ted.cmis.csiro.au/~dave/TR2000-83.ps.gz

[3] Lawrence Page, Sergey Brin, Rajeev Motwani and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. *Standford University Database Group*. http://dbpubs.stanford.edu:8090/pub/1999-66

5. Appendix

5.1. Selected responses from the pre-search questionnaire

Q1: How much experience have you had searching with WWW search engine?

Q5: When I search the WWW, I can usually find what I am looking for.

Q6: I always use the query box – keeping rephrase my queries until I find the right information.

Q7: I always browse web directory (e.g. Yahoo subject directory, etc) to get the information.

Q8-1: How often do you conduct searching for information about assignment/work related project?

Q8-2: How often do you conduct searching for information about shopping?

Q8-3: How often do you conduct searching for information about traveling?

Q8-4: How often do you conduct searching for information about medical/health?

Q8-5: How often do you conduct searching for information about government policy?

Q8-6: How often do you conduct searching for information about entertainment?

Q10: Please indicate your level of expertise with searching. (Novice(1).....Expert(7)).

Q11: Overall, for how many years have you been doing online searching? _____ years.

5.2. Post-search questionnaire

PS1: Was it easy to get started on this search?

PS2: Was it easy to do the search on this topic?

PS3: Are you satisfied with your search results?

PS4: Did you have enough time to do an effective search?

PS5: Did your previous knowledge help you with your search?

PS6: Have you learned anything new about the topic during your search?

(All above questions on a seven-point Likert scale.)

CWI at the TREC-2002 video track

Thijs Westerveld, Arjen de Vries, Alex van Ballegooij
CWI
PO Box 94079, 1090 GB Amsterdam
The Netherlands
{thijs,arjen,alexb}@cwi.nl

1 Introduction

We present a probabilistic model for the retrieval of multimodal documents. The model is based on Bayesian decision theory and combines models for text based search with models for visual search. The textual model, applied to the LIMSI transcripts, is based on the language modelling approach to text retrieval. The visual model, a mixture of Gaussian densities, describes keyframes selected from shots. Both models have proved successful on media specific retrieval tasks. Our contribution is the combination of both techniques in a unified model, ranking shots on ASR-data and visual features simultaneously.

Using this model, we tried to answer the following questions.

- Is it useful to identify important parts in query images?
- Can using (additional) query images from outside the search collection[1] help improve retrieval results?
- Does it help to have *multiple* image examples for a query, or are we better of using only *one* good example?
- Can a combination combined textual and visual query perform better than queries in a single modality?

Because of problems with the similarity measure we used in the submitted runs, we mainly report on post-hoc experiments on the TREC-2002 data in which we used a different measure. Both measures are discussed in Section 2, where we also present our retrieval model. Section 3 reports on the post-hoc experiments and Section 4 summarises our main findings. The official results can be found in appendix A.

2 Probabilistic Multimedia Retrieval

In a probabilistic retrieval setting, the goal is to find the document D^* with highest probability given a query Q:

$$D^* = \arg\max_i P(D_i|Q) = \arg\max_i \frac{P(Q|D_i)P(D_i)}{P(Q)} \tag{1}$$

Usually, (1) is used as a scoring function and a ranked list is returned rather than the one most probable document.

If we assume that all documents have equal prior probability, (1) reduces to the maximum likelihood (ML) criterion, which is approximated by the minimum KL-divergence between query model and document model: $D^* = \arg\min_i \mathrm{KL}[P_q(\mathrm{x})||P_i(\mathrm{x})]$.

$$\mathrm{KL}[P_q(\mathrm{x})||P_i(\mathrm{x})] = \int P(\mathrm{x}|D_q)\log\frac{P(\mathrm{x}|D_q)}{P(\mathrm{x}|D_i)}\mathrm{dx}$$

$$= \int P(\mathrm{x}|D_q)\log P(\mathrm{x}|D_q)\mathrm{dx} - \int P(\mathrm{x}|D_q)\log P(\mathrm{x}|D_i)\mathrm{dx},$$

[1] Throughout this document, we refer to the search collection used in the TREC-2002 video track as *the search collection*.

where $\mathbf{x}$ are feature vectors describing the documents.

The first integral is independent of D_i and can be ignored, thus

$$D^* = \arg\max_i \int P(\mathbf{x}|D_q) \log P(\mathbf{x}|D_i) \mathrm{d}\mathbf{x} \qquad (2)$$

Now suppose query and document models generate a mixture of textual features $\mathbf{x_t}$ and visual features $\mathbf{x_v}$:[2]

$$P(\mathbf{x}|D_i)) = P(\mathbf{x_t}|D_i)P(t) + P(\mathbf{x_v}|D_i)P(v).$$

We can then integrate over these different feature sets separately and arrive at the following ranking formula for multimodal retrieval [5].

$$\begin{aligned} D^* = \arg\max_i \, [P(t) \int_{\mathbf{x_t}} P(\mathbf{x_t}|D_q) \log P(\mathbf{x_t}|D_i) \mathrm{d}\mathbf{x_t} \\ + P(v) \int_{\mathbf{x_v}} P(\mathbf{x_v}|D_q) \log P(\mathbf{x_v}|D_i) \mathrm{d}\mathbf{x_v}] \end{aligned} \qquad (3)$$

2.1 Text Model

To describe the probability distributions of the textual terms, we take a language modelling approach to information retrieval [2]. Such a model operates on discrete signals (i.e. words), thus we can replace the integral from (3) by a sum. Moreover, the query model D_q is usually nothing more than the empirical distribution of the query, therefore we only need to sum over the words in the query. The document model is usually taken to be a mixture of foreground ($P(x_{t,j}|D_i)$) and background ($P(x_{t,j})$) probabilities for the query terms $x_{t,j}$, interpolated using mixing parameter λ (cf. Section 2.1.1). If our textual query consists of N_t terms $\mathbf{x_t} = (x_{t,1}, x_{t,2}, \ldots, x_{t,N_t})$ then the textual part of our ranking formula is the following.

$$D_t^* = \arg\max_i \frac{1}{N_t} \sum_{j=1}^{N_t} \log \left[\lambda P(x_{t,j}|D_i) + (1-\lambda) P(x_{t,j})\right] \qquad (4)$$

Using the statistical language modelling approach for video retrieval, we would like to exploit the hierarchical data model of video, in which a video is subdivided in scenes, which are subdivided in shots, which are in turn subdivided in frames. Statistical language models are particularly well-suited for modelling such complex representations of the data. We can simply extend the mixture to include the different levels of the hierarchy, with models for shots and scenes:[3]

$$\begin{aligned} \text{Shot}^* = \arg\max_i \frac{1}{N_t} \sum_{j=1}^{N_t} \log[\lambda_{\text{Shot}} P(x_{t,j}|\text{Shot}_i) + \\ \lambda_{\text{Scene}} P(x_{t,j}|\text{Scene}_i) + \lambda_{\text{Coll}} P(x_{t,j})] \\ \text{with } \lambda_{\text{Coll}} = 1 - \lambda_{\text{Shot}} - \lambda_{\text{Scene}} \end{aligned} \qquad (5)$$

The main idea behind this approach is that a good shot contains the query terms and is part of a scene having more occurrences of the query terms. Also, by including scenes in the ranking function, we hope to retrieve the shot of interest, even if the video's speech describes it just before it begins or just after it is finished. Depending on the information need of the user, we might use a similar strategy to rank scenes or complete videos instead of shots, that is, the best scene might be a scene that contains a shot in which the query terms (co-)occur.

2.1.1 Estimating Parameters

The features in the textual part of our model are simply the words themselves. For the textual part of our retrieval function (5), we only need to estimate foreground ($P(x_{t,j}|D_i)$) and background ($P(x_{t,j})$) probabilities. Both measures are estimated in the standard way, by taking the term frequency and document frequency respectively [2]. We used the TREC-2002 video search collection to find the optimal values for the mixing parameters: $\lambda_{\text{Shot}} = 0.090$, $\lambda_{\text{Scene}} = 0.210$, and $\lambda_{\text{Coll}} = 0.700$. Since we trained these parameters on the test collection. we cannot say anything about how well these numbers generalise across

[2] $P(t)$ and $P(v)$ are the prior probabilities of drawing respectively textual or visual features from a document; assumed uniform across documents.

[3] We assume each shot is a separate class and replace ω_i with Shot_i.

collections.[4] Yet, for each of the mixing parameters, there is quite a large range of values for which the scores are close to optimal. In this work we do not look into the stability of these parameters across collections, we are only interested in finding the optimal settings for this collection and evaluating the retrieval model with these optimal settings.

2.2 Image Model

We use a Gaussian Mixture Model for describing document densities [4].

$$P(\mathbf{x_v}|D_i) = \sum_{c=1}^{C} P(\theta_{i,c})\ \mathcal{G}(\mathbf{x_v}, \boldsymbol{\mu}_{i,c}, \boldsymbol{\Sigma}_{i,c}),$$

where C is the number of components in the mixture model, $\theta_{i,c}$ is component c of document model D_i and $\mathcal{G}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the Gaussian density with mean vector $\boldsymbol{\mu}$ and co-variance matrix $\boldsymbol{\Sigma}$:

$$\mathcal{G}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}\|\mathbf{x}-\boldsymbol{\mu}\|_{\boldsymbol{\Sigma}}}, \qquad (6)$$
$$\text{where } \|\mathbf{x}-\boldsymbol{\mu}\|_{\boldsymbol{\Sigma}} = (\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

and n is the length of the feature vector $\mathbf{x}$.

2.2.1 Bags of Blocks

Just like in our textual approach, for the query model, we can simply take the empirical distribution of the query samples. If a query-image $\mathbf{x_v}$ consists of N_v samples: $\mathbf{x_v} = (x_{v,1}, x_{v,2}, \ldots, x_{v,N_v})$ then $P(x_{v,i}|D_q) = \frac{1}{N_v}$. For the document model, we take a mixture of foreground and background probabilities, i.e. the (foreground) probability of drawing a query sample from the document's Gaussian mixture model, and the (background) probability of drawing it from any Gaussian mixture in the collection. In other words, the query image is viewed as a bag of blocks (BoB), and its probability is estimated as the joint probability of all its blocks. The BoB measure for query images then becomes:

$$D_v^* =$$
$$\arg\max_i \frac{1}{N_v}\sum_{j=1}^{N_v} \log\left[\kappa P(x_{v,j}|_i) + (1-\kappa)P(x_{v,j})\right], \qquad (7)$$

where κ is a mixing parameter and the background probability $P(x_{v,j})$ can be found by marginalising over all M documents in the collection:

$$P(x_{v,j}) = \sum_{i=1}^{M} P(x_{v,j}|D_i)P(D_i).$$

Again we assume uniform document priors ($P(D_i) = \frac{1}{M}$ for all i). In text retrieval, one of the reasons for mixing the document model with a collection model is to assign non-zero probabilities to words that are not observed in a document. Smoothing is not necessary in the visual case, since the documents are modelled as mixtures of Gaussians, having infinite support. Another motivation for mixing is to weight term importance: a common sample $\mathbf{x}$ (i.e., a sample that occurs frequently in the collection) has a relatively high probability $P(\mathbf{x})$ (equal for all documents), and therefore $P(\mathbf{x}|D)$ has only little influence on the probability estimate. In other words, relatively common terms and common blocks influence the final ranking only marginally.

2.2.2 Asymptotic Likelihood Approximation

A disadvantage of using the BoB measure is its computational complexity. In order to rank the collection given a query, we need to compute the posterior probability $P(\mathbf{x_v}|\omega_i)$ of each image block $\mathbf{x_v}$ in the query for each document ω_i in the collection. For evaluating a retrieval method this is fine, but for an interactive retrieval system, optimisation is necessary.

An alternative is to represent the query image, like the document image, as a Gaussian model (instead of by its empirical distribution as a bag of blocks), and then compare these two models using the KL-divergence. Yet, if we use Gaussians to model the class conditional densities of the mixture components, there is no closed-from solution for the visual

[4] Obviously, the official runs have used different mixing parameter values, see Appendix A.

part of the resulting ranking formula (3). As a solution, Vasconcelos assumes that the Gaussians are well separated and derives an approximation, ignoring the overlap between the mixture components: the asymptotic likelihood approximation (ALA) [4]. The ALA is the measure we used in our official TREC-2002 runs, (see Appendix A). However, in post hoc analysis, we found that one of the assumptions underlying the ALA is not plausible for the collection at hand and, moreover, using it decreases performance compared to the BoB measure (for details see [5]). In the remainder of this work we will concentrate on the BoB measure.

2.2.3 Estimating Parameters

For estimating the parameters of the Gaussian mixture model, we used the EM algorithm [1]. We described a document as a set of samples, where each sample is described by a number of DCT coefficients in the YCbCr colour space[5]. Then we used EM to fit a mixture of 8 Gaussian (for details see [5]). Finally, we described the position in the image plane of each component as a 2D-Gaussian with mean and covariance computed from the positions of the samples assigned to this component. We evaluated different values for mixing parameter κ on the TREC-2002 video search collection and found the optimal value: $\kappa = 0.9$.

3 Experiments

Fully automatic creation of queries from topic descriptions was not required in this year's video track. However, there was a distinction between manual and interactive runs. In an interactive run a user can interact with a system to locate relevant shot. In a manual run a user has one go at creating a query from a topic descriptions and then submits this query to the system to retrieve relevant shots. All our runs are manual runs in which we experimented with different ways of creating queries from topic statements. In the following subsections, we investigate the following questions:

- Is it useful to identify important parts in query images?
- Can using (additional) query images from outside the search collection help improve retrieval results?
- Does it help to have *multiple* image examples for a query, or are we better of using only *one* good example?
- Can a combination combined textual and visual query perform better than queries in a single modality?

3.1 Selecting Query Images

In general, it is hard to guess what would be a good example image for a specific query. If we look for shots of the *Golden Gate bridge*, we might not care from what angle the bridge was filmed, or if the clip was filmed on a sunny or a cloudy day; visually however, such examples may be very different (Figure 1). If a user has presented three examples and no additional information, the best we can do is try to find documents that describe all example images well. Unfortunately, a document may be ranked low even though it models the samples from one example image well, as it may not explain the samples from the other images.

For each topic, we computed which of the example images would have given the best results if it had been used as the only example for that topic. We compared these *best example* results to the *full topic* results in which we used all available visual examples. In the *full topic* case, the set of available topics was regarded as one large bag of blocks. We ranked documents by their probability of generating all blocks in all query images. For the single image queries in the *best example*, we used all samples from the single visual example to rank documents.

Since it is problematic to use multiple examples in a query, we wanted to see if it is possible to guess in advance what would be a good example for a specific

[5] We use the first 10 coefficients from the Y channel and the two DC coefficients of the Cb and the Cr channels.

Figure 1: Visual examples of the Golden Gate bridge.

topic. Therefore, we hand-picked for each topic a single representative from the available examples and compared these *manual example* results to the other two result sets.

The results for the different settings are listed in Table 1. A first thing to notice is that all scores are rather low. When we take a closer look at the topics with higher average precision scores, we see that these mainly contain examples from the search collection. In other words, we can find similar shots from within the same video, but generalisation is a problem.

The fact that using the best image example outperforms the use of all examples shows that indeed combining results from different visual examples can degrade results. Looking at the results, manually selecting good examples seems a non-trivial task, but the drop in performance is partly due to the generalisation problem. If one of the image examples happens to come from the collection it scores high. If we fail to select that particular example, the score for the manual example run drops. Simply counting how often the manually selected example was the same as the best performing example, we see that this was the case for 8 out of 13 topics.[6]

	full topic	best example	manual example
vt075	.0038	**.2438**	**.2438**
vt076	**.4854**	.4323	.1760
vt077	.0000	.0000	.0000
vt078	.0000	.0000	.0000
vt079	.0000	**.0040**	.0000
vt080	.0048	**.0977**	**.0977**
vt081	.0000	.0000	.0000
vt082	**.0330**	.0234	.0234
vt083	.0000	.0000	.0000
vt084	**.0046**	**.0046**	**.0046**
vt085	.0000	.0000	.0000
vt086	.0053	**.0704**	**.0704**
vt087	.0000	.0000	.0000
vt088	.0046	**.0069**	**.0069**
vt089	.0000	.0000	.0000
vt090	.0000	**.0305**	**.0305**
vt091	.0095	**.0095**	**.0095**
vt092	.0003	**.0106**	.0000
vt093	.0006	**.0006**	.0000
vt094	.0021	**.0021**	**.0021**
vt095	.0000	.0000	.0000
vt096	**.0323**	**.0323**	**.0323**
vt097	.1312	**.1408**	.0000
vt098	.0000	**.0003**	**.0003**
vt099	.0000	.0000	.0000
MAP	.0287	.0444	.0279

Table 1: MAP for Full Topics, Best Examples and Manual Examples

3.2 Selecting Important Regions

In last year's video track, we saw that query articulation, i.e. the manual identification of important parts in a query image, can help improve retrieval results [3]. We also noticed that this requires an enormous effort from a user. In our probabilistic setting, selecting important (and coherent) regions is much

[6] We ignored the topics for which there is only one example and the ones for which none of the examples retrieved relevant documents.

easier. After building a query-model (like we build document models) a user can simply select one or more meaningful components from the query-model. In retrieval, we can then use only the Bag of Blocks corresponding to the selected component(s). For example in Figure 2a, we selected the components that together form the US flag. Similarly, we can indicate we want multiple parts to be present in the target shots, e.g. *boat* and *water* and *sky* (Figure 2b). Note that even though the union of the sets of samples is in this case the full image, this differs from simply taking the using all samples as a query. If the full image were used, we would have looked for shots with relatively few water samples; the selection of components compensates for that and looks for documents that explain all 3 concepts equally well.

From each of the query images, we selected meaningful components and we used the corresponding samples as queries. If we take a look at the individual components and their results, we see that the components are often homogeneous in colour and/or texture and that results are often meaningful (Figure 3) or, if there is little semantics in the component, at least visually similar (Figure 4). It is not clear yet how this can be used for highly specific queries like the video track queries.

3.3 Using Query Examples from Outside the Collection

In Section 3.1, we argued that selecting the right query image is important. On the one hand therefore, one would like to expand a query to have as many different query images as possible. On the other hand, we saw that it is difficult to combine multiple examples in one query (Section 3.1). We investigate whether using (additional) examples from outside the collection can improve retrieval effectiveness. We expect that this is not the case; in previous experiments [5, 3] we saw that we can only find relevant shots if the query images are highly similar to the relevant shots, i.e. if they are from the same collection and preferably from the same video.

First of all, we had a look at the original examples provided by NIST. Most, if not all, of the video examples in this set come from either the search collection itself, or the highly comparable[7] feature train or feature test set. We found that the topics that contributed most to our MAP score were the ones with examples from the search collection. If we remove videos from which shots are used as examples from the relevance judgements, our MAP score for a purely visual run (using full examples for all queries) drops from .0287 to .0029; purely visual runs from other groups show a similar drop in performance. This indicates that visual retrieval systems are able to locate the query examples in the collection, but generalisation seems problematic. Furthermore, the best examples as reported in table 1 are mainly video examples from either the search collection or the comparable training data. Only for three topics, the best scoring example was an image example from outside these collections. Yet, for these three topics no video examples were available.

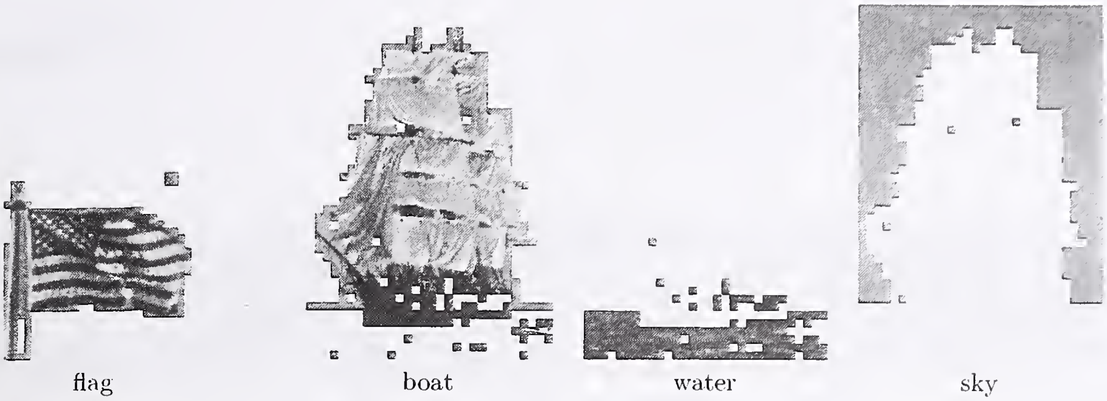
We experimented with query expansion by adding additional example images found using Google image search[8]. We manually created short queries from the topic descriptions and submitted these to Google image search. From the result list we selected images based that we thought were good examples for the topic. This way we expanded topics with up to 7 additional image examples. We ran these new examples as queries against the collection and recomputed the best scoring examples for each topic. For 5 out of 25 topics none of the examples retrieved any relevant documents. The best scoring examples for the remaining 20 topics were video examples in 12 cases and image examples from Google in 8 cases. Clearly, if we try more examples we have a better chance of having a good example among them, yet the problem remains how to combine multiple examples or how to identify a good example without knowledge of the relevant documents in the collection.

3.3.1 Combining Textual and Visual runs

We combined textual and visual runs using our combined ranking formula (3). Since we had no data to estimate the parameters for mixing textual and visual information we used $P(t) = P(v) = 0.5$. For the

[7] In fact, these are distinct subsets of one larger collection.

[8] http://images.google.com



Find additional shots with one or more US flags flapping

a

Find shots with one or more sailboats, sailing ships, clipper ships, or tall ships - with some sail(s) unfurled

b

Figure 2: Selecting components from images

textual part we tried both short and long queries, for the visual part we used full queries and best-example queries. Table 2 shows the results for combinations with the BoB measure. We also experimented with combinations with the ALA measure, but we found that in the ALA case it is difficult to combine textual and visual scores, because they are on different scales (see also Appendix A). The BoB measure is closer to the KL-divergence and, on top of that, more similar to our textual approach, and thus easier to combine with the textual scores.

For most of the topics, textual runs give the best results, however for some topics using the visual examples is useful. This is mainly the case when either the topics come from the search collection or when the relevant documents are outliers in the collection. This illustrates how difficult it is to search a generic video collection using visual information only. We only succeed if the relevant documents are either highly similar to the examples provided or very dissimilar from the other documents in the collection (and therefore relatively similar to the query examples). When both textual and visual runs have reasonable scores, combining the runs can improve on the individual runs, however, when one of them has inferior performance, a combination only adds noise and lowers the scores.

4 Conclusions

We presented a probabilistic framework for multimodal retrieval in which textual and visual retrieval models are integrated seamlessly and evaluated the framework using the search task from the TREC-2002 video track. We found that even though the topics were specifically designed for content-based retrieval, and relevance was defined visually, a textual search outperforms visual search for most topics. The main conclusion in this work is that visual retrieval using the presented model for specific queries does not generalise very well. The model could retrieve shots that are highly similar to the query examples (i.e. shots from the same video). but other similar shots were found mostly by coincidence, because they happened to have for example the same colour sky or grass. For more general queries, the model seems useful. When we select a single component from an example, results are intuitive, i.e. visually similar. It is unclear how this helps in retrieving relevant documents for highly

Topic	Tshort	Tlong	BoBfull	BoBbest	BoBfull +Tshort	BoBfull +Tlong	BoBbest +Tshort	BoBbest +Tlong
vt075	.0000	.0082	.0038	.2438	.0189	.0569	.2405	**.3537**
vt076	.4075	.6242	.4854	.4323	.5931	**.7039**	.5757	.6820
vt077	.1225	**.5556**	.0000	.0000	.0000	.0000	.0000	.0000
vt078	.1083	**.2778**	.0000	.0000	.0000	.0000	.0000	.0000
vt079	.0003	.0006	.0000	.0040	.0003	.0000	**.0063**	.0050
vt080	.0000	.0000	.0048	**.0977**	.0066	.0059	.0845	.0931
vt081	.0154	**.0333**	.0000	.0000	.0037	.0000	.0000	.0000
vt082	.0080	.0262	.0330	.0234	.0181	**.0335**	.0145	.0210
vt083	**.1669**	**.1669**	.0000	.0000	.0962	.0962	.0078	.0078
vt084	**.7500**	**.7500**	.0046	.0046	.6875	.6875	.6875	.6875
vt085	.0000	.0000	.0000	.0000	.0000	.0000	.0000	.0000
vt086	.0554	.0676	.0053	.0704	.0536	.0215	**.0791**	.0600
vt087	**.0591**	.0295	.0000	.0000	.0052	.0003	.0052	.0003
vt088	**.0148**	.0005	.0046	.0069	.0052	.0046	.0069	.0069
vt089	**.0764**	**.0764**	.0000	.0000	.0503	.0503	.0045	.0045
vt090	.0229	.0473	.0000	.0305	.0006	.0075	.0356	**.0477**
vt091	.0000	.0000	**.0095**	**.0095**	.0000	.0086	.0000	.0086
vt092	.0627	**.0687**	.0003	.0106	.0191	.0010	.0078	.0106
vt093	**.1977**	.1147	.0006	.0006	.0099	.0021	.0071	.0012
vt094	.0232	**.0252**	.0021	.0021	.0122	.0036	.0122	.0036
vt095	**.0034**	.0021	.0000	.0000	.0008	.0012	.0011	.0010
vt096	.0000	.0000	**.0323**	**.0323**	.0161	.0161	**.0323**	**.0323**
vt097	.1002	.0853	.1312	.1408	.1228	**.1752**	.1521	.1474
vt098	**.0225**	.0086	.0000	.0003	.0068	.0000	.0004	.0003
vt099	**.0726**	.0606	.0000	.0000	.0000	.0000	.0000	.0000
MAP	.0916	**.1212**	.0287	.0444	.0691	.0750	.0784	.0870

Table 2: Average precision per topic, for Textual runs. BoB runs and combined runs

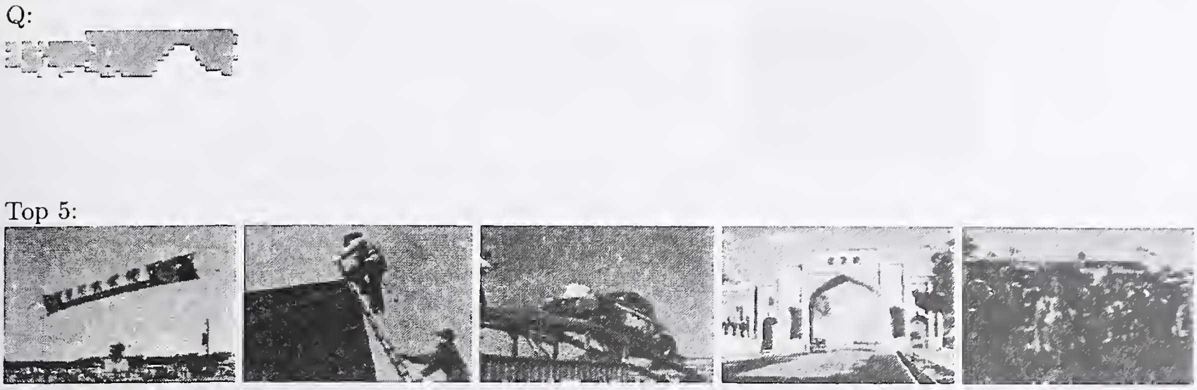


Figure 3: Top 5 results for a homogeneous query with clear semantics ('Sky')

specific topics like the video track topics, but it helps in gaining insight in the models performance. In future work, we will further investigate the influence of individual components on retrieval results. In addition, we intend to look at how incorporating different sources of additional information (e.g. contextual frames, the movement in video or user interaction) can help improve results across collections. Combining multiple examples in one query is still problematic, but combining textual and visual runs seems possible using the presented framework. When one of the runs is poor, a combined run, including the noise, is less effective than the single best run. However, when the individual runs have reasonable scores, combining them improves retrieval effectiveness.

References

[1] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, series B*, 39(1):1–38, 1977.

[2] D. Hiemstra. *Using language models for information retrieval.* PhD thesis, Centre for Telematics and Information Technology, University of Twente, 2001.

[3] The Lowlands team. Lazy users and automatic video retrieval tools in (the) lowlands. In *The 10th Text Retrieval Conference (TREC-2001)*, 2002.

[4] N. Vasconcelos. *Bayesian Models for Visual Information Retrieval.* PhD thesis, Massachusetts Institut of Technology, 2000.

[5] T. Westerveld, A. P. de Vries, A. van Ballegooij, F. M. G. de Jong, and D. Hiemstra. A probabilistic multimedia retrieval modela and its evaluation. *EURASIP Journal on Applied Signal Processing, special issue on Unstructured Information Management from Multimedia Data Sources*, 2003.

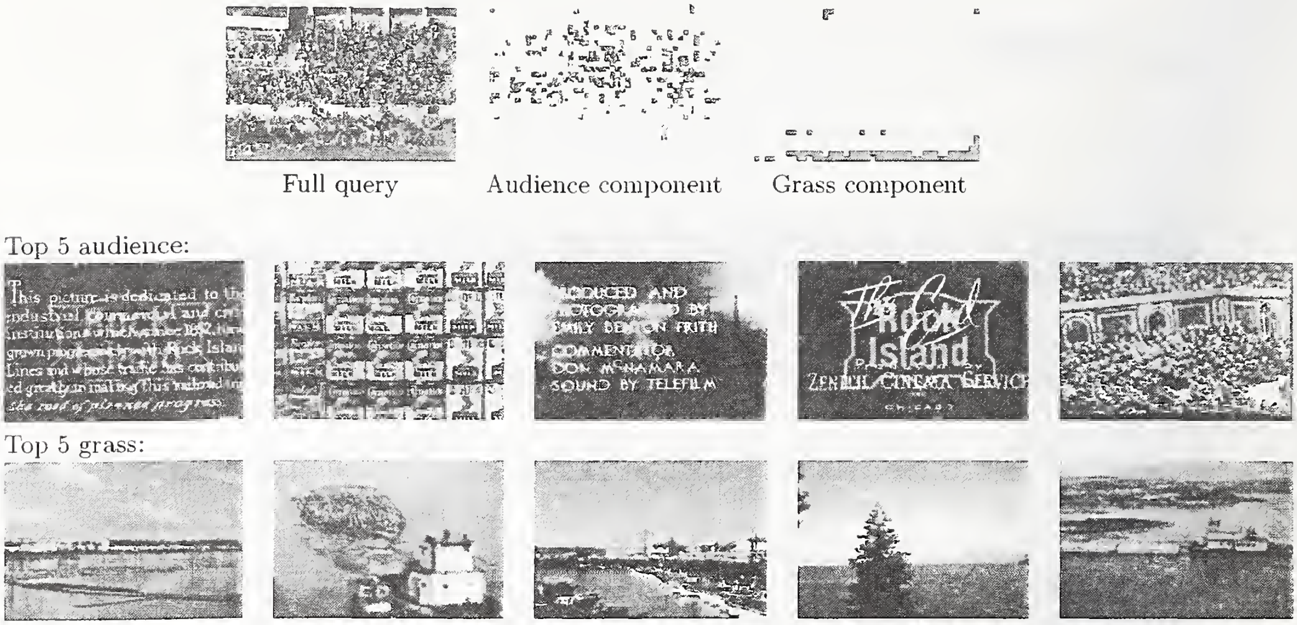
A Official Results

In the official runs, we used the Asymptotic Likelihood Approximation (see section 2.2.2). We distinguished between the NIST images (the visual examples from the official topics) and Google images (additional examples we found with manual query expansion using Google and submitted four runs:

run1 Text only.

run2 Text + NIST images.

run3 Text + selected components from NIST images.



Full query

Audience component

Grass component

Top 5 audience:

Top 5 grass:

Figure 4: Top 5 results for homogeneous queries with unclear or false semantics

run4 Text + selected components from both NIST and Google images.

The text model's mixing parameters have been optimized using the TREC-2001 corpus, giving $\lambda_{\text{Shot}} = 0.015$, $\lambda_{\text{Scene}} = 0.135$, and $\lambda_{\text{Coll}} = 0.850$. For run3 and run4, we manually selected *important* components from the query model (cf. Section 3.2). In all runs that involved visual examples, we computed a single new (8 component) Gaussian mixture model from all available visual blocks and we used that model in our ALA ranking formula. The results for the official runs and for the same runs after fixing some bugs[9] are shown in Table 3. We see that also with the ALA measure text only results are by far the best (run 1). Combinations that also used visual information scored lower, not only on MAP, but also on average precision for each individual topic. In contrast to our findings with the BoB measure we were not able to combine textual and visual information properly.

runName	MAP
run1	.0917
run2	.0016
run3	.0022
run4	.0038
run1 fixed	.1212
run2 fixed	.0082
run3 fixed	.0137
run4 fixed	.0069

Table 3: Official results and same runs after bug fix.

[9] A normalisation error in the training of the models and exchanging a few videos from the search and feature detection collections.

Dublin City University Video Track Experiments for TREC 2002

Paul Browne, Csaba Czirjek, Cathal Gurrin, Roman Jarina, Hyowon Lee, Seán Marlow, Kieran Mc Donald, Noel Murphy, Noel E. O'Connor, Alan F. Smeaton, Jiamin Ye

Centre for Digital Video Processing, Dublin City University, Glasnevin, Dublin 9, Ireland
Alan.Smeaton@computing.dcu.ie

Abstract

Dublin City University participated in the Feature Extraction task and the Search task of the TREC-2002 Video Track. In the Feature Extraction task, we submitted 3 features: Face, Speech, and Music. In the Search task, we developed an interactive video retrieval system, which incorporated the 40 hours of the video search test collection and supported user searching using our own feature extraction data along with the donated feature data and ASR transcript from other Video Track groups. This video retrieval system allows a user to specify a query based on the 10 features and ASR transcript, and the query result is a ranked list of videos that can be further browsed at the shot level. To evaluate the usefulness of the feature-based query, we have developed a second system interface that provides only ASR transcript-based querying, and we conducted an experiment with 12 test users to compare these 2 systems. Results were submitted to NIST and we are currently conducting further analysis of user performance with these 2 systems.

1. Introduction

This year Dublin City University took part in two of the three tasks in the Video Track: Feature extraction task and Search task. In Section 2 we present the Feature extraction task that we conducted (Speech, Instrumental Sound and Face), the methods used for each feature, and our results. In Section 3 we present the Search task – specifically the interactive video retrieval system that we developed for the task and the experiment procedures and our results. The system is a variation of the Físchlár Digital Video System with an XML-based architecture that uses an MPEG-7 compliant video description. The system provides a web-based user interface from which a user can compose a query based on all the 10 features and the ASR transcript text. We used the system in the interactive search task with 12 test users who each conducted searches for the 25 topics provided by the Video Track.

2. Feature Extraction Task

Ten features were listed for the feature extraction task and we extracted three of the features ourselves: Speech, Instrumental Sound and Face.

2.1 Speech Extraction

The task here is to recognise a shot as having a human voice uttering words. In our approach, speech characteristics were derived from the volume (energy) contour of the frequency-limited audio signal. There are some properties that distinguish speech from other signals. Roughly, speech exhibits an alternating sequence of 3 kinds of sounds that have different acoustic properties: i) *vowels and vowel-like sounds* – longer tonal quasi-periodic segments with high energy, which is concentrated in lower frequencies: ii) *fricative consonants* – noise-like short segments with lower volume and spectral energy distributed more toward the high frequencies; iii) *stop consonants* – short silent segments followed by a very short transition noise pulse. These three kinds of sounds alternate and form the regular syllabic structure of speech and therefore strong temporal variations in the amplitude of speech signals can be observed.

Our speech detector does not use an audio signal waveform as the input data, rather it utilises information taken directly from the MPEG-1 audio encoded bitstream. Thus a time-consuming decoding process is not required, and in addition information from audio signal analysis (e.g. subband filtering, volume estimation) already stored in the MPEG encoded bitstream, is utilized. The MPEG audio layer-II frame consists of 1,152 samples: 3 groups of 12 samples from each of 32 subbands. A group of 12 samples in each subband gets a bit allocation and, if this is not zero, a scalefactor. Scalefactors are weights that rescale samples so that they fully use the range of the quantizer. The encoder uses a different scalefactor for each of the three groups of 12 samples only if necessary. By definition the scalefactors carry information about the maximum level of the signal in each subband. Thus, the volume contour of the overall audio signal can be estimated by the summation of the scalefactors over all subbands. In the case of a frequency-limited signal, this summation is done only over given subbands.

2.1.1 Procedure for Speech Extraction

Our approach was based on the measurement of the duration and the rate of the energy peaks of the audio signal. The method was first introduced in [1] where theoretical background and results of preliminary studies can be found. For the TREC task, the method had to be slightly modified. By trial examination of various parts of video recordings from the TREC *Feature Development Collection*, it was decided that at least 7 of the low frequency subbands must be included in the processing. In some cases, the first subband (frequencies up to 0.7 kHz) was excluded from analysis. The procedure of signal analysis and processing for speech detection is described below.

- Each video file was demultiplexed, and the MPEG-1 audio layer II bitstreams were stored in separate files (MP2 files). Then, only the scalefactors of the first 7 subbands were extracted from the MP2 files.
- First, silence detection was carried out. An energy level of the signal was determined by the superposition of all relevant scalefactors. The frames in which the level was below the threshold, were assigned as silent frames.
- In the case of speech detection, the envelope of the band-limited signal was estimated by summing relevant scalefactors from the 2nd to the 7th subbands only. This procedure was followed by a 5th order median filtering to avoid rapid random changes in the amplitude.
- For analysis, a sliding window was used with a window length of 3.9 seconds and a 1.3 second shift (i.e. 2/3 overlap).
- Energy peaks were extracted by a simple thresholding procedure. Two low-level features were chosen for speech detection. These are: i) *Lm* – the duration of the widest peak within the analysis window (segment); ii) *R* – the rate of peaks (number of peaks in the analysis window). Each segment was assigned to speech or non-speech by using a simple rule-based decision procedure. This process has been discussed in greater detail in [1]. All the MP2 audio frames corresponding to an analysed segment were given a relevance value of '1' in the case of a speech segment, and the value '0' otherwise.
- The silent parts of the signal longer than 1.5 seconds were assigned as non-speech signal.
- Final speech feature measures for the standard video shots were determined by averaging the relevance values over all the audio frames within each video shot.

2.1.2 Evaluation and Test Results

For evaluation, we submitted the top 1,000 standard video shots ranked according to the highest possibility of detecting the speech feature. The results of our runs are summarised in Table 1.

	Our Results	Maximum	Median
Average precision	0.710	0.721	0.656
Precision at 100 results	1.000	1.000	0.980
Precision at 1000 results	0.987	0.997	0.944

Table 1. Speech test results (compared with maximum and median values among TREC participants)

2.2 Instrumental Sound Extraction

This feature characterises a sound produced by one or more musical instruments. Henceforth this feature is referred to as the *music feature*. The music feature detection task is a much more challenging task than the speech detection task. Unlike speech, musical sounds are very difficult to define due to their great variety and uncertain nature. However, musical signals have some unique characteristics, which may help to discriminate them from other sounds. Music tends to be composed of a multiplicity of tones, each with its own distribution of higher harmonics. The energy contour has usually a much smaller number of "peaks" and "valleys" and it shows either very little change over a period of several seconds (e.g. classical music) or strong long term periodicity due to exact rhythm (e.g. dance music).

For this TREC task we developed a method that is an extended version of the method we already used for speech detection (see section 2.1). Two other low-level features were incorporated into the system to improve discrimination between musical sounds and other environmental sounds. They are: rhythm and harmonicity. We believe that most of the sounds produced by instrumental music have harmonic structure of spectra unlike noise-like environmental sounds. The importance of rhythm detection has been recently discussed in [2].

2.2.1 Procedure for Instrumental Sound Extraction

The first two features *Lm* and *R* (duration and rate of the energy peaks) were computed in the same way as in section 2.1.1. In addition, the rhythm (or pulse) metric *Pm* and harmonic ratio H were computed.

- Similar to [2], the rhythm metric is expressed by the following procedure.
For each of the first 7 subbands, the normalised autocorrelation function $\overline{R}$ were computed.

$$\overline{R}_k(t)=R_k(t)/R_k(0),\quad R_k(t)=\sum_n e_k(n)\cdot e_k(n+t),$$

where $e_k(n)$ is the subband energy contour (or envelope) in the k-th subband, without its DC component. The subband energies were estimated directly from the scalefactors of the MPEG-1 layer II bitsream. For rhythm analysis, a sliding window with a 4/5 overlap was used. We searched $\overline{R}$ within the analysed window over the interval corresponding to time $t = 0.2$ - 1.75 seconds to find peaks. We set $p(j)$ to the value of the highest peak in the j-th subband. Then we defined the feature rhythm metric P_m as

$$P_m = \max\{p(k)\},\quad k=1,2,...7,\qquad 0<P_m<1$$

The higher the value of P_m, the greater amount of rhythmicality in the signal.

- The harmonicity ratio defines the degree of harmonicity of an audio signal. We computed it in accordance with the MPEG-7 description schema [3]. By definition, the harmonicity ratio is the ratio of harmonic power to total power. It was computed by the following procedure:

At first, comb filtering is applied

$$r(k)=\sum_j s(j)s(j-k)\Bigg/\left(\sum_j s(j)^2 * \sum_j s(j-k)^2\right)^{0.5}$$

where s is the sequence of PCM samples of the band-limited signal. Only the 2nd subband was used for the computation. Thus the sampling frequency was f_2=44.1kHz / 32. Index k was changed up to the value corresponding to the maximum expected fundamental period (around 20 ms). The Harmonicity ratio H was determined as the maximum value of $r(k)$ for each frame. $H = 1$ for a purely periodic signal, and it will be close to 0 for white noise.
- These four low-level features *Lm, R, Pm* and *H* were used as inputs for a heuristic rule-based classifier. The relevance for each analysed segment was computed as a weighted sum of these features. The weights and thresholds for the classifier were determined by trial and error examination of various parts of video recordings from the TREC *Feature Development Collection*. Similarly as in the case of speech detection, silence detection was performed. For the silent parts, which were longer than 1.5 seconds, the music relevance was set to zero.
- Final music/instrumental sound feature measures for the standard video shots were determined by averaging the relevance scores over all the audio frames corresponding to the given video shot.

2.2.2 Evaluation and Test Results

For evaluation, we submitted the top 300 standard video shots ranked according to the highest possibility of detecting the speech feature. The results are summarised in Table 2. We reach the highest precision at 100 results among TREC participants, but the precision at 1,000 results and the average precision are very low because we submitted only 300 results for evaluation. Since we identified much more relevant shots than we submitted for judgement, we have re-calculated precision and average precision for our 1,000 top ranked shots and these unofficial results are also shown marked with * in Table 2. The unofficial average precision is 0.494 which shows that our method performs very well.

	Our results	Maximum	Median
Average precision (official) Average precision (unofficial)	0.222 0.494 *	0.637	0.347
Precision at 100 results	0.970	0.970	0.845
Precision at 1000 results (official) Precision at 1000 results (unofficial)	0.281 0.650 *	0.877	0.667

Table 2. Instrumental sound test results (compared with maximum and median values among TREC participants)

2.3 Face Extraction

As presented in [4], the colour of human skin falls into a relatively narrow band of the colour space. Many colour models have been used in pre-processing the input image, in order to locate potential human presence. We know [5] that normalised RGB, YUV, HSV, CIEL etc. can be used for this purpose. In this task, we decided to detect skin-like pixels using a similar approach to [6], updating the filtering technique based on the available *Feature Development Collection*.

2.3.1 Procedure for Face Extraction

Due to the binary nature of classification, the output skin-mask will be populated with isolated skin-like pixels, i.e. noise. In order to address this undesirable effect, we applied a morphological open-close filtering. After this operation, we expect to obtain homogeneous areas of connected pixels. Having the skin-map, we want to group together connected pixel areas into regions. Therefore a connected component labelling was performed, which gave the number of regions used in further processing. Even applying morphological filtering to the skin-map, regions with a small number of pixels may occur. To reduce the number of false candidate regions, areas with the number of pixels less than $N = 625$ were ignored. We have chosen this threshold based on the assumption that no face could be detected by this method having a size smaller than 25x25 pixels. Horizontal and vertical strips, which are less likely to contain a human face, were also ignored. These regions were detected by having a huge difference between width and height, with the requirement that the smaller dimension does not exceed 25 pixels.

Assuming that the human face has an approximately elliptical shape, for each connected component (region) the best-fit ellipse was calculated based on moments [7]. Unfortunately, many other objects in a visual scene have the same colour characteristics as the human skin, or other object(s) are merged with the face (i.e. hands, background wall, etc.). An intermediate step in the processing chain consists of an iterative partitioning of regions having "irregular" shape. This means breaking a region S into component convex sub-regions Sn, n being the number of sub-regions, by applying K-means clustering.

The detection task is based on principal component analysis of the remaining skin patches. Given a collection of test images, we constructed a face space for discriminating the remaining candidate regions. The measure of "faceness" of the input sample relies on the reconstruction error, expressed as the difference between the input image and its reconstruction using only the M eigenvectors corresponding to the highest eigenvalues:

$$\varepsilon^2 = \| x - \bar{x} \|^2 \text{ (DFFS)}$$

The distance from face space (DFFS) indicates how well the test image can be approximated by the most significant eigenvectors spanning the eigenspace. The distance between the projected input image and the mean face image in the feature space is given by the norm of the principal component vector. Since the variance of a principal component vector y_i is given by its associated eigenvalue λ_i, the squared Mahalanobis distance measure d^2 gives a measure of the difference between the projection of the test image and the mean face image of the training set {x}:

$$d^2 = \sum_{i=1}^{M} \frac{y_i}{\lambda_i}$$

where y_i are the projection coefficients and λ_i are the associated eigenvalues. Therefore d^2 can be expressed as the distance in face space (DIFS). Given these two distances a combined error criterion was used:

$$e = d^2 + c\varepsilon^2 .$$

where $e \in [0,1]$, and c is a suitable constant value. As we work with digital video, a confidence measure is attached to each continuous video shot, meaning the level of certainty that a face occurs. Because of time constraints, the above algorithm processes each 10-th frame in sequence. The confidence measure for a shot is expressed as the average confidence value of each processed frame within the shot.

2.3.2 Evaluation and Test Results

We submitted for evaluation the highest ranked 300 shots and our results are summarised in Table 3. Within the first 100 shots, precision was 0.53 which was similar to the median. Our precision at 1000 is low due to the fact that we only submitted the top 300 results.

	Our Results	Maximum	Median
Average precision	0.154	0.613	0.166
Precision at 100 results	0.530	0.990	0.540
Precision at 1000 results	0.114	0.312	0.221

Table 3. Face detection results (compared with maximum and median values among TREC participants)

3. Interactive Search Task

For the Search task, we conducted an interactive search experiment with test users. For this we developed an interactive video searching/browsing system which is a variation of our Físchlár system, and conducted a lab experiment using the system with 12 test users. The hypothesis we were testing was that ASR + features searching outperforms ASR-only searching in our controlled environment.

3.1 System Description

The system we used for the search task is a variation of the Físchlár Digital Video System [8]. an online video system which has been operational for 3 years within the University campus and which we used for the interactive search task in the previous year's Video Track where we compared 3 different keyframe browsers. Currently the Físchlár system has a XML-based architecture and uses MPEG-7 compliant video description internally. While having the same underlying architecture as the Físchlár system, the system we tailored for this year's search task is more sophisticated in its search mechanism and user interface, as it provides various query methods for users based on the feature extraction data, some of which is our own (Face, Speech, Music) as well as donated features namely Indoor, Outdoor, People, Landscape and Text Overlay from IBM, and Monologue and Cityscape from Microsoft Research Asia. The system also allows the users to execute text queries over the test collection, based on the donated Automatic Speech Recognition (ASR) transcript provided by LIMSI. This transcript used an American English broadcast news transcription system and is described in [11].

3.1.1 System Architecture

Figure 1 shows the components of the Físchlár system. The system uses an internal XML description as its core element (in the centre of Figure 1). When a user submits a query via the web-based interface, the web application processes it and sends the query detail to the logic element governing the search engine (see Section 3.1.2). The search engine sends back the retrieved results with relevance scores to the XML generator, which generates the necessary XML descriptions dynamically, to be transformed by appropriate XSL stylesheets to render HTML and SVG for display back on the user's web browser.

The queries that a user generates can be composed of any, some, or all of the following elements:

- *Feature listing of the required features*, there were ten features in all (excluding ASR transcript) and the user could select any of these features for inclusion in the query. However, there were some interface restrictions placed on users, for example, a user could not specify in a query that shots be *both* Indoor and Outdoor.
- *Query text*, which would be matched against the ASR transcript. While the system supported querying based on features alone, our findings indicated that all users relied on ASR text when constructing queries.
- *An identifier of the video within which to search*, if the query was at the shot level. Our system supported both searching for videos and searching for shots within a particular video, hence the support for specifying a shot identifier within certain queries.

3.1.2 Retrieval and Weighting Scheme

In order to support search and retrieval over the video data we developed a search server, which was designed to support both ASR-only querying and ASR + feature querying for both the shots and the videos as a whole (the lower half of Figure 1). Each user's search session is essentially a two-phase process. The first phase was to generate a ranked list of videos in response to a user query, where each of the 176 videos were scored and ranked before being returned in decreasing rank order to the user. The user could then select one of the videos (usually one of the higher ranked) for shot-level examination, which was the second phase. Shot-level examination results in the search server producing a ranked listing of shots from within the selected video that match the user's query, the same query that originally generated the ranked list of videos. Our ranking technique was developed without using the TREC topics (no training data) and thus it was not developed specifically to provide high retrieval performance on this particular corpus and associated queries.

The ASR transcripts for each shot (donated by LIMSI) were pre-processed to remove stopwords and then stemmed using Porter's algorithm. When a user submits search term(s) as part of a query, these search terms undergo the same process. Each shot was represented by the ASR transcript text associated with the particular shot while each video was represented by the combination of all ASR text associated with all the shots that comprise the particular video. This required the utilisation of two conventional (text-only) search engines based on BM25 with the following parameter values; advl = 900, b = 0.75, k1 = 1.2 and k3 = 1000 which were set according to the best performance achieved on the WT2g collection from TREC-8 [9]. The scores for each query were normalised to be in the range [0..1] to allow for easier combination with the feature scores. We note that for any additional experimentation it would be advantageous to tune BM25 parameters to best-fit ASR content.

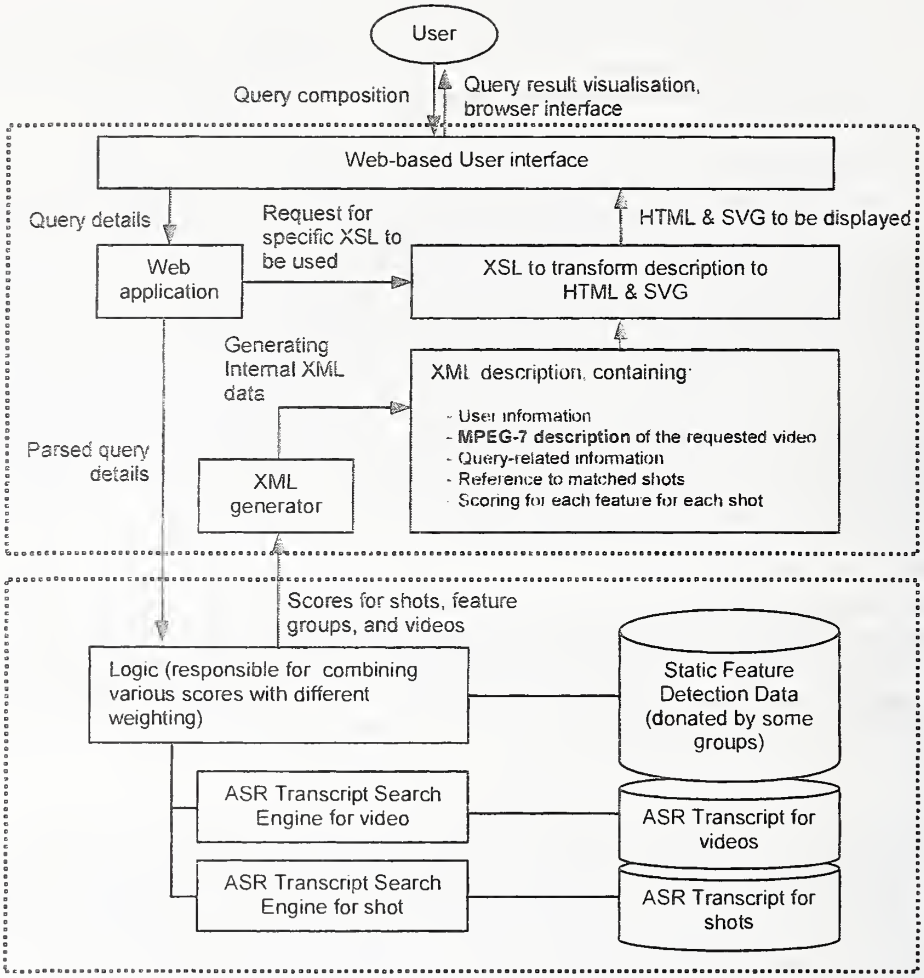

User
Query composition
Query result visualisation, browser interface
Web-based User interface
Query details
Request for specific XSL to be used
HTML & SVG to be displayed
Web application
XSL to transform description to HTML & SVG
Generating Internal XML data
XML description, containing:
- User information
- MPEG-7 description of the requested video
- Query-related information
- Reference to matched shots
- Scoring for each feature for each shot
Parsed query details
XML generator
Scores for shots, feature groups, and videos
Logic (responsible for combining various scores with different weighting)
Static Feature Detection Data (donated by some groups)
ASR Transcript Search Engine for video
ASR Transcript for videos
ASR Transcript Search Engine for shot
ASR Transcript for shots

Figure 1: System architecture of Físchlár-TREC2002

3.1.2.1 Search and Retrieval of Video Units

Recall that the first phase of a user's search was to generate a ranked list of videos in response to the query. Each video is represented by an overall feature weight for each of the 10 features, which was generated by calculating the aggregate scores for each feature from each shot within that video and then dividing these aggregate scores by the total number of shots in the video.

Without having carried out a sampling of the accuracy of the feature detection we were using and given that our features originated from 3 separate participating groups (with large variations in average feature confidence) we normalised the weights of each feature so that no one feature would outweigh any other feature due to differences in confidence levels. In addition, we weighted each feature's influence based on its usefulness as an aid to distinguishing between different videos. For this we utilised a variation of the conventional text-ranking methodology *idf*. This allowed us to increase the weighting of features that are better able to support distinguishing between relevant and non-relevant videos. In this way we weighted features that were better able to distinguish between videos higher than features that occurred in all or virtually all videos.

In response to a user's query, a ranked list of videos is returned to the user for further consideration. The overall rank for each video was based on linear combination of required (as specified in the query) feature influence along with the ASR search score. The influence of the ASR text in the video retrieval phase was weighted 4 times higher than any of the other ten features – this being our best-guess parameter reflecting our belief that the ASR transcript feature would be the primary method of ranking videos.

3.1.2.2 Search and Retrieval of Shot Units

Upon the user selecting a ranked video from the first phase for shot level examination, the query used to generate the ranked list of videos was augmented with an identifier of the chosen video and then sent to the search server in order to rank shots from within that particular video. The algorithm used to rank shots within a selected video is similar to that used to rank the videos with the following exceptions: the normalisation of feature weights for shots was calculated at a shot level as opposed to the video level; the weighting of each feature's influence was also calculated at the shot level and the ASR text scores were weighted at twice that of features in order to allow features to play a greater role in shot ranking than in video ranking.

When a user is examining a video at the shot level the ranking outlined above is only one of six sorting options available to the user. These six options discussed in 3.1.3 are chronological (using the weighting above for the SVG timeline as shown in Figure 2), combined (also using the above weighting but for shot ordering) and four feature groupings as discussed in 3.1.3 which do not use the shot level ranking described above.

3.1.3 Web Interface with XSL

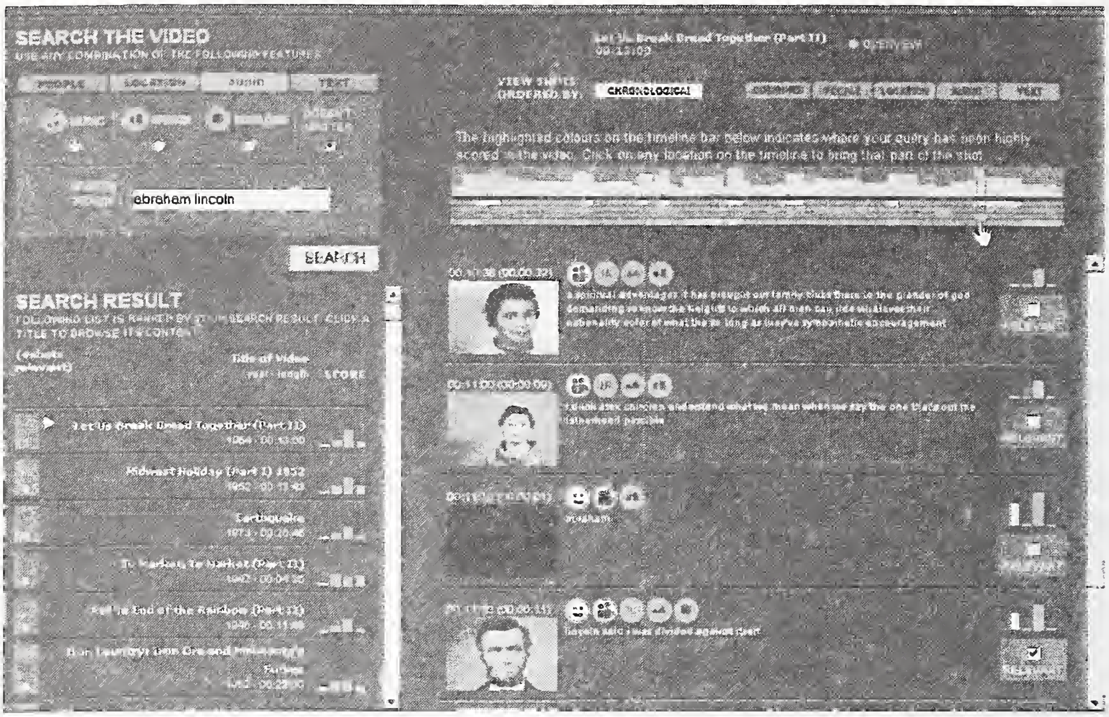
Having an internal XML-based architecture allowed us to clearly separate the presentation of data on the user interface from how the system works internally, significantly helping the system development process where software engineering and interface design can happen separately once the full XML format has been agreed.

For displaying on a web browser, XSL (eXtensible Stylesheet Language) has been extensively used on top of XML descriptions. XSLs were created when designing the interface (at design time), and used in conjunction with internal XML descriptions at user search time. XSLs transformed the internally generated XML video descriptions into 2 different formats based on a user's request – HTML and SVG (Scalable Vector Graphics). HTML is used to render most of the information display on the browser, including video listing with icons, score bars, ASR transcript, and other elements. This includes interactive elements such as ToolTips and JavaScript to enhance interaction. SVG is used to render a timeline on a chronologically displayed shot listing, plotting an indication of the matching status of the four feature groups against the user's query. Transforming to HTML means that any conventional web browser can be used to display the system's interface, though a SVG plug-in is required for viewing the SVG timeline and an Oracle plug-in is also required for streamed playback of video. Figure 2 shows a screen shot of the interface.

A user specifies her query on the query panel at the top left of the screen. All 10 features and ASR transcript query are grouped into 4 broad groups with distinctive colours associated with each. These are:

- **People:** Face(s), Group of People
- **Location:** Indoor, Outdoor, Cityscape, Landscape
- **Audio:** Music, Speech, Monologue, and ASR transcript search box
- **Text:** Text Overlay

Note that we included ASR transcript search as part of the third group. The query panel is organised by tabs, showing only one of the 4 feature groups at a time. In this way we expected to provide a simple and intuitive query screen to the users (4 features groups rather than 11 features) and the consequent retrieval result visualisation also makes use of the 4 grouping's colour schemes. The user specifies her query by clicking on the radio buttons for each feature, indicating if the feature is required or not. Some features have been intentionally made to be mutually exclusive (e.g. Indoor and Outdoor cannot be specified at the same time). Clicking on the SEARCH button triggers retrieval (see section 3.1.2.1) and the result is displayed below the query panel, as a list of video programmes in a ranked order. For each video programme, score bars are presented indicating the relative scores of the 4 feature groups used in the user's query. Clicking on a title of the video in the list displays the content of the video on the right side and executes shot-level retrieval within that video. Initially an *overview* of the video programme is presented with the title. textual description and about 30 keyframes selected by equal time distance within the video. The user can further search for the wanted shots if they wish by clicking on the CHRONOLOGICAL button. which presents all of the chronologically ordered individual shots with the detected features, a keyframe. an ASR text portion, as well as score bars for the 4 feature groups. This is shown in Figure 2. Each of the shot entries also displays small round icons for the features detected for that shot, when their confidence value is above a threshold. At the top of the shot list in the chronological view, an SVG timeline is presented displaying the query matching status for each of the 4 feature groups as well as the combined score. The highlighted segment in the timeline indicates the part of the video that has matched against the query. The user can then click on the timeline to jump to the corresponding shot in the shot list below.



The highlighted colours on the timeline bar below indicates where your query has been highly scored in the video. Click on any location on the timeline to bring that part of the shot

Figure 2: Web-based user interface of Físchlár-TREC2002

The user can re-order the shots by combined score (see section 3.1.2.2), or by any of the 4 feature groups by clicking on the buttons beside CHRONOLOGICAL button, allowing quick access to shots in relation to a subpart of the query she specified. At any point while browsing, the user could click on a keyframe to start streamed playback of the video from that shot onwards, and this allowed the user to clarify if indeed a shot is relevant to a search topic. If a user finds a shot that she believes to be relevant to the search topic, she ticked a checkbox in each shot entry to indicate this, and the initial search result list (on the left of the screen) updated showing the number of shots she has indicated as relevant in the video programme.

3.2 Experimental Procedure

For our experiment we created a second version of our system, which supported only ASR transcript searching and not feature-based searching, in order to compare this to the full feature system. Our aim was to compare the two systems to see if the 10 semantic features aided retrieval more than simply relying only on the text-based transcript searching. We also observed the interactive behaviour of users of the system in order to get additional feedback from test users.

Twelve people participated as test users, 10 postgraduate students and 2 summer intern staff in the School of Computer Applications within the University. All had advanced levels of computer knowledge and familiarity with web-based searching, each conducting some form of online searching on a daily basis. Each of the 12 test users conducted all of the 25 query topics provided by NIST, one by one. The test users were divided into 2 groups, one group conducting the 25 topics in one order. and the other using the same topics but in reverse order. Six users used the full-feature system with all 10 features and the ASR transcript text searchable (3 forward and 3 in reverse order). and another 6 users used the system that had ASR transcript-only searching (also 3 forward and 3 in reverse order).

Each test user was seated in front of a desktop PC with headphones in a computer lab, and completed the first part of the questionnaire. We used the questionnaire developed over several years by the TREC Interactive track [10]. The questionnaire included pre-test questions, short post-topic questions. and post-test questions. which each of the users filled in at each stage of the testing. After a brief introduction, test users used a series of web pages which presented each topic, including the audio/image/video examples which form part of the topic descriptions. Users read, viewed, and played the examples that accompanied the topic and then conducted their search. Users were given 4 minutes for searching each topic and whenever a shot was located that the searcher thought answered the topic, they indicated this by checking the relevant box beside the shot entry (see Figure 2). At the end of the 4 minutes, users filled in a short post-topic questionnaire. and waited to be asked to start the next topic. The time taken to read the topic and examine the associated media elements was included in the four minute allocation per query. At the end of 12th topic, the users took 10-15 minute break for light refreshments. After the break the next 13 topic searches continued, finishing with the post-test questionnaire. All individual users' interactions were logged by the system, and the results of users' searching were collected and from these results four runs were submitted to NIST for evaluation.

3.3 Submitted Runs

As mentioned above, we submitted four runs to NIST. These were the following:

1. *Full-feature system with all users (I_B_DCUTrec11B_1),* where the selected shots of all users that used the full-feature system were aggregated (combined together) and this aggregated listing was sent as our first run to NIST.
2. *ASR transcript-only system with all users (I_B_DCUTrec11C_2),* where the selected shots of all users who used the ASR transcript-only system were aggregated and the aggregated shot listing was submitted.
3. *User with highest number of shots selected in the Full-feature system (I_B_DCUTrec11B_3),* where the results of the individual user who selected the highest number of shots using the full-feature system was submitted as our third run.
4. *User with highest number of shots selected in the ASR transcript-only system (I_B_DCUTrec11C_4),* where the results of the individual user who selected the highest number of shots using the ASR transcript-only system were submitted as our fourth and final run.

Note that the run 1 (I_B_DCUTrec11B_1) and 2 (I_B_DCUTrec11C_2) cannot be directly compared with the runs from other groups as these aggregate 6 individual users' results.

3.4 Results of the Experiments

Figure 3 illustrates the average precision of each of our four runs. As can be seen the figures illustrate that no significant benefit in retrieval performance was found when the features were used in the search and retrieval process, which is the hypothesis we were testing. If we examine the user performance for the user with highest recall then it seems that the ASR+features interface aids the user more than the ASR–only interface, but with such a small number of users we can not say this with confidence.

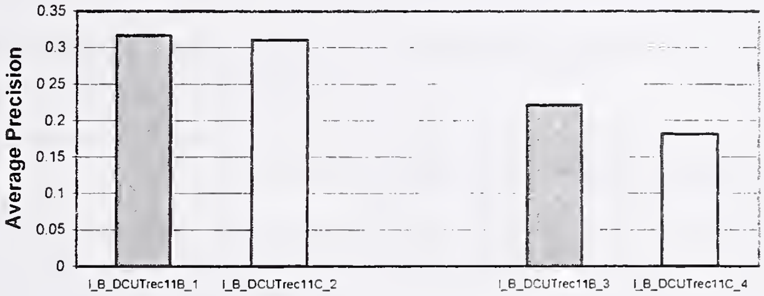


Figure 3: Average Precision of our four submitted runs

Post-experiment examination of the results of each user show us that while the feature interface worked well for some users, others had difficulties using it and the variance in recall attained by users of the feature interface was almost double that of the ASR-only interface.

Our observations after running the experiment suggest that a user's primary method of searching was using the ASR transcript, with features being used in addition when their inclusion seemed reasonable. This is clearly illustrated by examination of the two topics for which all 12 of our participants failed to find any relevant documents. Both of these topics (75 and 91) required search terms that were not in our chosen ASR transcript ('Rickenbacker' and 'parrot'), so when the ASR transcript could not aid retrieval, features were found to be of no benefit.

4. Conclusion

From the feature extraction task we observe that due to the nature of our approach to face detection, our system ran into difficulties operating on grayscale videos and slightly coloured material. Obvious improvements could be made by using a different approach which does not rely on skin colour segmentation. Our results for Speech extraction showed our method worked very well. If we consider our full 1,000 identified shots of our unofficial run for

Instrumental Sound extraction instead of our 300 submitted ones of the official runs, our performance compared favourably with other participants' results.

From the search task we find ourselves unable to come to any significant conclusions yet about the benefit of incorporating features into the retrieval process. More work needs to be done on methods of combining the features with the ASR transcript. In addition, the experiment has illustrated to us the need to provide users with query-focussed overview as opposed to our overviews (see section 3.1.3) that used 30 temporally selected keyframes. Observations of the user experiments suggest that some users will not examine a video at the shot level if the overview does not show relevant keyframes regardless of the video's ranked position. Finally, our system seemed to operate very well as a browsing tool supporting search, however we do wonder whether a user needs to go through video level ranking before examining shots. Further experimentation into direct shot-based ranking across videos would answer whether this supports faster resource discovery or reduces the high variability of user performances we observed in our experiment.

Acknowledgements: We would like to thank the groups from IBM, Microsoft Research Asia and LIMSI who each provided some of the feature detection output that we used in our search submissions. The support of the Enterprise Ireland Informatics Research Initiative is also gratefully acknowledged.

References

[1] Jarina, R., Murphy, N., O'Connor, N. and Marlow, S., "Speech-music discrimination from MPEG-1 bitstream", in Kluev, V.V., and Mastorakis, N.E. (eds.). *Advances in signal processing, robotics and communications*, WSES Press, 2001, pp. 174-178.

[2] Jarina, R., O'Connor, N. and Marlow, S, "Rhythm Detection for Speech-Music Discrimination in MPEG Compressed Domain", *Proc. of the IEEE 14th International Conference on Digital Signal Processing DSP 2002*, Santorini, Greece, July 2002, pp. 129-132

[3] ISO/IEC JTC 1/SC 29/WG 11, "Information Technology — Multimedia Content Description Interface — Part 4", Audio, March 2002.

[4] Yang, M.H. and Ahuja, N., "Detecting Human Faces in Color Images" *Proc. IEEE Int'l Conf. Image Processing*, October 1998, pp. 127-239.

[5] Yang, M.H., Kriegman, D.J. and Ahuja, N., "Detecting Faces in Images: A Survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**(1), January 2002, pp. 34-58.

[6] Sobottka, K. and Pittas, I., "A novel method for automatic face segmentation, facial feature extraction and tracking ", *Signal Processing: Image Communication* **12**, 1998, pp.263-281.

[7] Jain, A.K., "*Fundamentals of digital image processing*", Prentice-Hall, NJ, 1989.

[8] Lee, H. and Smeaton, A.F., "Designing the user interface for the Físchlár Digital Video Library". *Journal of Digital Information, Special Issue on Interactivity in Digital Libraries*, 2(4), May 2002.

[9] Savoy, J. and Rasolofo, Y., "Report on the TREC-9 Experiment: Link-based Retrieval an Distributed Collections". *Proceedings of the 9th Annual TREC Conference*, November 16-19, 2000.

[10] TREC Interactive Track. Available online at URL: http://www-nlpir.nist.gov/projects/t11i/t11i.html (last visited October 2002).

[11] Gauvan, J.L., Lamel, L. and Adda, G., "The LIMSI Broadcast News Transcription System", *Speech Communication*. 37(1-2):890198, May 2002.

Semantic Feature Extraction using Mpeg Macro-block Classification

Fabrice Souvannavong, Bernard Merialdo and Benoît Huet
Département Communications Multimédias
Institut Eurécom
2229, route des crêtes
06904 Sophia-Antipolis - France
(Fabrice.Souvannavong, Bernard.Merialdo, Benoit.Huet)@eurecom.fr

Abstract

In this paper, we present some first results in the extraction of semantic features from video sequences. Our approach is based on the classification of Mpeg DCT macro-blocks. Although it is clear that using macro-blocks imposes severe restrictions on the analysis accuracy of the image, it has the advantage of avoiding the complete decoding of the Mpeg stream. Our objective is to evaluate the quality of the Semantic Feature Extraction that can be obtained with this direct approach, to serve as a comparative baseline with more elaborate approaches.

Keywords: *Semantic classification, Discrete Cosine Transform, Gaussian Mixture Models, Compressed Domain.*

1 Introduction

The large amount of visual information, carried by video documents as well as still images, requires efficient and effective indexing and search tools [2, 6]. The U.S. Institute of Standards and Technology sponsors the serie of TREC [1] 2002 conferences to promote progress in content-based retrieval from digital video. Our work takes place in this context where we focus on the feature extraction task; video shots should be classified into the high level semantic concepts *indoor, outdoor, cityscape, landscape, text overlay, face* and *people*.

To extract relevant features, the content should in principle be decoded first. Since this operation is time consuming, especially when a large video database should be processed, feature extraction directly from the compressed domain would be particularly interesting by providing fast and reliable information analysis and selection tools. Lots of work have been conducted to achieve image or video analysis [3], however only few researchers have given solutions to this challenging task with limited decoding of the mpeg stream [10, 4].

In this paper, we propose to extract semantic features from 16 by 16 pixels DCT macro-block classification. We have distinguished two types of features in the TREC set, the **region-level** features like *face* and *text overlay* and the **frame-level** features like *indoor, outdoor, cityscape, landscape* and *people* that require elementary concepts like *building, greenery, sky* and *water* to be detected.

The next section details the supervised classification process via Gaussian Mixture Models [9, 7] of macro-blocks. Then we explain how the final decision is taken by introducing new elementary concepts to describe frame-level semantics. Finally, we will outline future improvements.

[1] TREC is a series of conferences which high-level goal is the investigation of content-based retrieval from digital video.
See http://www-nlpir.nist.gov/projects/t2002v/t2002v.html

2 Macro-block Classification

In the context of supervised classification, three steps are involved: feature extraction and representation, class modelisation and parameter estimation, finally classification with respect to decision rules.

In our approach, features are directly provided by the video stream after parsing since we work only on I-frames, which are encoded somehow like jpeg pictures. These frames are composed of macro-blocks that contain 6 DCT blocks, 4 for Y color component, 1 for U and 1 for V i.e. 4:2:0 video format. We can represent a DCT macro-block by a vector of size 64 corresponding to the zigzag scan of the DCT block coefficients and then make the concatenation of the 6 vectors to obtain the feature vector of the whole region. Since the first DCT coefficients are the most important i.e: to eye sensitivity and noise, the feature space dimension is simply reduced to 60 by truncation. Moreover coefficients are scaled with respect to their importance in order to increase the sensitivity of the classifier to important components and at the same time to slightly improve the initialisation of the training algorithm, which is usually obtained via k-means algorithm as explained in the next subsection.

We assume a mixture model to describe the distribution of macro-blocks for each class, and specifically a multi-dimensional Gaussian distribution. Gaussian models can capture the characteristics of a macro-block, while modeling the variation due to motion or lighting conditions. Moreover in [5], E.Y. Lam and J.W. Goodman have proven that the distribution of macro-block DCT coefficients can be well approximated by a Gaussian *when the variance is constant*; in the classification situation, the latter hypothesis is more or less true and mixtures should compensate it. So the probability density function can be written as follows:

$$\text{For } X \in C_i, P(X \mid \Phi_i) = \sum_j \alpha_j p_j(X)$$

where $\alpha_i \in \Re, \Phi_i = (\mu_j, \sigma_j)$ and $p_j(X) \sim \mathcal{N}(\mu_j, \sigma_j)$

The GMM parameters α_j, μ_j and σ_j are estimated using the traditional Expectation-Maximization algorithm [1] which is initialized with a classical k-means algorithm. In our current experiments, we also make the hypothesis that feature vector components are independent, thus σ_i is a diagonal matrix, or that only color components of the same frequency are correlated, thus σ_i is a matrix diagonal by block. Finally the choice of the number of mixtures is simply achieved by looking at the test set loglikelihood evolution of the EM algorithm for various mixture numbers. It should not increase to much in order to avoid data overfitting.

Given an unlabeled macro-block X, the maximum a posteriori rule:

$$\hat{C} = arg \max_i P(\Phi_i \mid X)$$

gives an estimation of the class it belongs to. The posterior probabilities can be expanded by Baye's rule:

$$P(\Phi_i \mid X) = \frac{P(X \mid \Phi_i) P(\Phi_i)}{P(X)}$$

finally,

$$P(\Phi_i \mid X) \propto P(X \mid \Phi_i)$$

since we assume the *equiprobability* of classes and vectors.

However, it is possible that a macro-block does not belong to any predefined class. Thus we introduce for each model i a minimum bound $-mb_i$ for the loglikelihood which is selected to eliminate 10% of the training data set. Of course there is a trade-off to find between precision and recall, see figure 1. Finally the decision rule can be written:

$$\hat{C} = arg \max_i \{P(X \mid \Phi_i) \mid -\log(P(X \mid \Phi_i)) \leq mb_i\}$$

3 Feature detection

The presented classification method allows to detect **region-level** features only. In our previous work [8] we have underlined that macro-blocks could not carry **frame-level** semantic information but succeed well in providing a lower level semantic. Thus a heuristic two-step hierarchy, depicted in figure 2, was introduced to detect **frame-level** concepts via additional elementary semantics. The hierarchy contains three kinds of elements:

- Elementary concepts at the leaves of the hierarchy that are perceivable from macro-blocks,

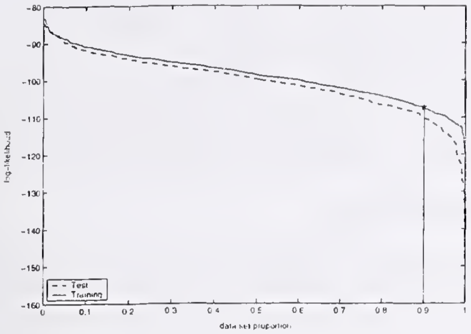


Figure 1: Threshold selection

- Higher-level semantics on the upper part of the graph that are difficult to extract directly, but can be induced by a combination of lower-level features.

- Trec concepts, enclosed in boxes, that are spread in either of the previously stated categories.

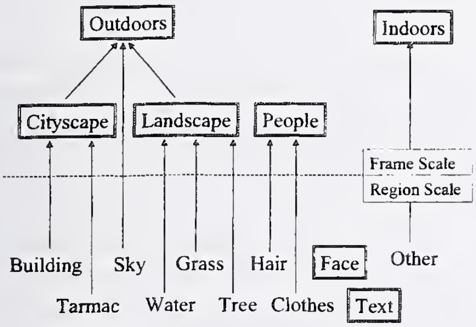


Figure 2: Concepts hierarchy.

The detection of features present in one shot is finally achieved with respect to the following procedure:

1. Classify all macro-blocks of the shot into elementary concepts with respect to preliminary trained Gaussian Mixtures,

2. Compute a detection score for each feature.

The detection score of the feature i whose elementary childrens are J is simply defined by:

$$Ds_i = \sum_J P(j) \text{ where } j \in J$$

$$P(j) = \frac{\text{Number of macro-blocks with label j}}{\text{Total number of macro-blocks in the shot}}$$

It represents the posterior probability of a feature to be in the given shot. Finally, for each feature, shots are ordered by decreasing detection score.

4 Experiments

Nine video sequences were randomly selected in the development set in order to create training and test samples. Some Macro-blocs of these sequences were labeled with **region-level** concepts; half to perform the training of semantic classes and half to evaluate models. The fastidious annotation task was achieved over 232 frames and table 1 gives a summary of the accomplished task.

We have finally modeled classes by fifteen gaussian mixtures and truncated the space dimension to six by fifteen features. This values reveal to be a good compromise between performance and complexity. For the same reasons, we have approximated the co-variance matrix to a diagonal and not diagonal by block matrix, see table 2 that emphasizes the small improvement acquired by using a diagonal by block co-variance matrix.

Finally figures 3 and 4 show the performance of our method thanks to the assesor's judgement provided by TREC. To evaluate the feature extraction task, we have represented the classical precision and recall curves for the four Trec features *outdoors, cityscape, text and face*. Encouraging results were obtained for *outdoors* and *cityscape* features, however we were expecting better results for *text* and *face* features since they are relevant at the macro-block level and the development analysis was forecasting good classification capacities, see table 2. Several explanations can be envisaged: heterogenous sizes of training sets leading to overtrained and undertrained models and too few training variety conducting to restricted models (for example, no cartoon sequences were used to train models). The results we obtained using a *single framework* for all visual features, are closely comparable to submitted runs of other labs. In particular we get

Selected sequences	00616	01859	06085a	08131b	08261	08325	16683	19567b	35435b
Nb of selected frames	46	18	14	38	42	26	21	17	10
Nb of selected blocks	4647	1464	2133	2745	6200	3549	1522	1879	1401

Features	text	skin	clothes	sky	tree	building	grass	tarmac	hair	water	ground	total
Number	1472	4097	7158	4636	1839	2550	558	639	857	297	1437	25540

Table 1: Summary of the manual annotation.

Features	text	skin	clothes	sky	tree	building	grass	tarmac	hair	water	ground
Diagonal by block											
precision	38	26	23	49	22	12	34	17	10	58	40
recall	75	56	45	88	69	45	66	84	44	68	81
Diagonal											
precision	33	30	22	41	18	12	26	12	6	19	30
recall	75	53	42	83	64	36	67	77	43	80	78

Table 2: Precision and recall during the development of the low level features.

surprisingly good ranking in text ovelay detection.

5 Conclusion

We have presented a method based on DCT information of macro-blocks to detect Trec visual features from video shots in a *single framework*. Since macro-blocks carry only local information, a heuristic hierarchy was introduced to build the final decision rule at the **frame-level** and **region-level**. In gereral this evaluation is encouraging knowing the small extract of the development set used. In future works we plan to investigate methods to automatically elaborate the hierarchy. This will set up a complete probabilistic framework to detect features from low level observations and a more realistic manual annotation at the shot level will be required to train models.

References

[1] J. A. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. ICSI-TR, 1997.

[2] S.-F. Chang, W. Chen, H. Meng, H. Sundaram, and D. Zhong. A fully automated content-based video search engine supporting spatiotemporal queries. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 8, pages 602–615, 1998.

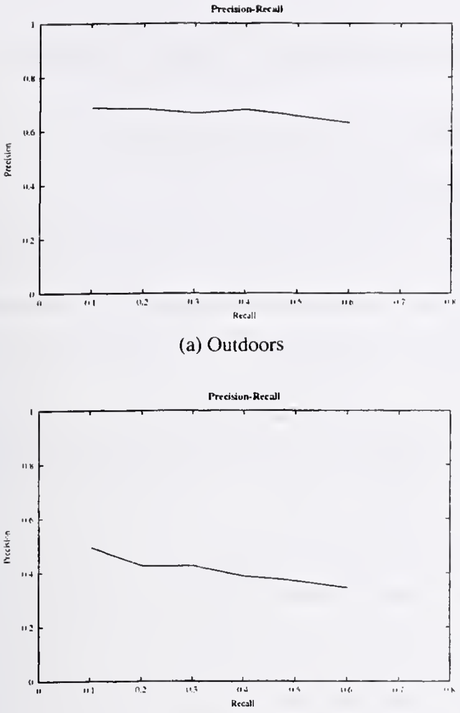
[3] S.-F. Chang and H. Sundaram. Structural and semantic analysis of video. In *ICME*, 2000.

[4] A. Girgensohn and J. Foote. Video classification using transform coefficients. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3045–3048, 1999.

[5] E. Y. Lam and J. W. Goodman. A mathematical analysis of the dct coefficient distribution for images. In *IEEE Transactions on Image Processing*, volume 9, pages 1661–1666, October 2000.

[6] A. Pentland, R. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases. In *SPIE Storage and Retrieval for Image and Video Databases*, february 1994.

[7] M. Saeed, W. Karl, T. Nguyen, and H. Rabiee. A new multiresolution algorithm for image segmentation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 5, pages 2753–2756, 1998.



(a) Outdoors

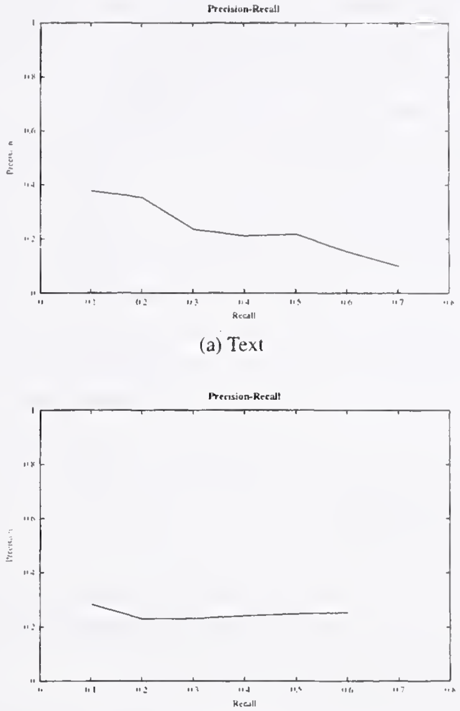
(b) Cityscape

Figure 3: Classification Evaluation

[8] F. Souvannavong, B. Merialdo, and B. Huet. Classification semantique des macro-blocs mpeg dans le domaine compresse. In *Compression et Representation des Signaux Audiovisuels*, pages 235–238, 2003.

[9] J. Verbeek, N. Vlassis, and B. Kr. Greedy gaussian mixture learning for texture segmentation. In *Workshop on Kernel and Subspace Methods for Computer Vision*, 2001.

[10] H. Wang and S.-F. Chang. A highly efficient system for automatic face region detection in mpeg video. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 7, pages 615–628, August 1997.



(a) Text

(b) Face

Figure 4: Classification Evaluation

FDU at TREC2002: Filtering, Q&A, Web and Video Tasks

Lide Wu, Xuanjing Huang, Junyu Niu, Yingju Xia, Zhe Feng, Yaqian Zhou

Fudan University, Shanghai, China

This year Fudan University takes part in the TREC conference for the third time. We have participated in four tracks of Filtering, Q&A, Web and Video.

For filtering, we only participate in the sub-task of adaptive filtering. A novel method is presented, in which a winnow classifier from the description and narrative fields is constructed, and then utilized to assist our previous adaptive filtering system.

A novel approach to confidence sorting, which is based on Maximum Entropy, is proposed in our Question Answering system. The rank of individual answer is determined by several weighted factors, and the confidence score is the product of the exponent of the weights of every factors. The weight of every factor is assigned during the training of previous questions.

To return highly relevant key resources for web retrieval, we modified our original search system to make it return higher precision result than before. First, we proposed a novel search algorithm to get a base set of highly relevant documents. Then special post-processing modules are used to expand and re-sort the base set.

This year we tried a fast manifold-based approach to face recognition in the Video Search Task. It can be used when there are only few different images of a specific person and runs fast. Experiment shows that applying this step will make the face recognition 5-fold faster and with almost no decreasing of performance.

1. Filtering

Our research focuses on how to make use of the narrative and description fields of each topic. Experiment results have shown that these two fields are very important and the proper exploitation of them can enhance the filtering system's performance greatly.

In this section, we will first introduce the training stage of our adaptive filtering system in details, and then the adaptive filtering stage; some experiment results are also given.

1.1 Training of adaptive filtering

Figure 1.1 shows the architecture of the training stage, which includes topic processing, feature vectors extracting and initial threshold setting.

1.1.1 Topic processing

By examining the words in the description and narrative of each topic carefully, we believe that smart use of these words will bring notable gain. Therefore, we construct a winnow classifier [Littlestone88] from these two fields to assist our adaptive filtering system.

First, we remove the function words from these two fields. Here function words including stop word and those words such as "relevant", "irrelevant", "documents" which bear no content in the topic description. Then we initialize a winnow classifier for each topic using the remaining words and assigning each word with equal weight. After that, we adjust the winnow item's weight during training. The adjustment procedure can be described as below:

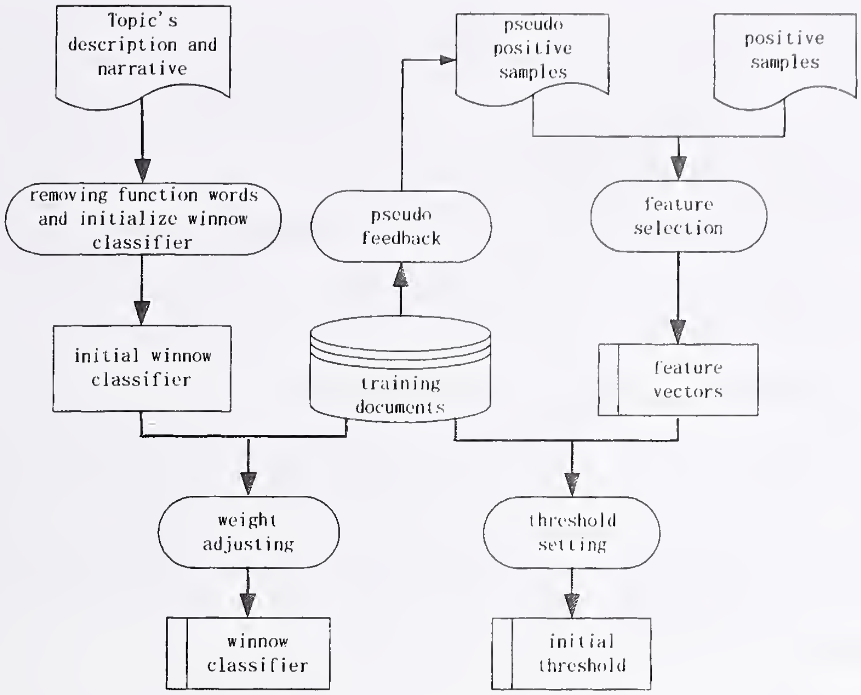


Figure 1.1 Architecture of training stage

If one of the positive samples of each topic (3 documents per topic) has not been retrieved by the winnow classifier, we promote weights of the words occur in this positive sample by a coefficient of 1.5.

If an irrelevant document has been retrieved by the winnow classifier, we demoted the words occur in this document by a coefficient of 0.8.

After that, we set threshold for each winnow classifier through the whole training set.

1.1.2 Feature selection

Since the total number of all words is very large and it costs much time in similarity computation, we decide to select some important words from them. First, we carry out morphological analysis and stopword removing. Then we compute the *logarithm Mutual Information* between remaining words and topics:

$$\log MI(w_i, T_j) = \log\left({P(w_i \mid T_j)} \Big/ {P(w_i)} \right) \tag{1.1}$$

Where, w_i is the ith word and T_j is the jth topic. Higher logarithm Mutual Information means w_i and T_j are more relevant. P(w_i) and P(w_i/T_j) are both estimated by *maximal likelihood method.*

For each topic, we select those words with logarithm Mutual Information higher than 3.0 and occurs more than once in the relevant documents. Logarithm Mutual Information is not only used as the selection criterion, but also as the weight of feature words.

1.1.3 Similarity Computation

The similarity between the profile and training documents is computed by the cosine formula:

$$Sim(d_i, p_j) = Cos\theta = \sum_k d_{ik} * p_{jk} \Big/ \sqrt{(\sum_k d_{ik}^2)(\sum_k p_{jk}^2)} \,. \qquad (1.2)$$

Where, p_j is the profile of the jth topic and d_i is the vector representation of the ith document. d_{ik}. the weight of the kth word in d_i, is computed as such: $d_{ik} = 1 + \log(tf_{ik} * avdl/dl)$, where tf_{ik} is the frequency of the kth word in the ith document , *dl* is the average number of different tokens in one document, *avdl* is the average number of tokens in one document.

1.1.4 Creating initial profile and Setting initial threshold

Each topic's feature vector is the weighted sum of feature vector from positive (relevant) documents and feature vector from pseudo relevant documents with the ratio of 1: 0.25. The pseudo relevant documents are acquired during pseudo feedback procedure, which uses the similarity as selection metric. Those documents that have highest similarity and do not occur in the positive documents are regard to be pseudo relevant.

After combining the positive and pseudo-positive feature vectors, we obtain the initial profile and then set the initial thresholds to get the largest value of T11SU or T11F.

1.2 Adaptive stage

Figure 1.2 shows the architecture for the adaptive stage. While filtering the input document stream, we first use the winnow classifier to make the initial decision. If winnow classifier has retrieved a document, the system will compute the similarity between this document and the feature vector and make final decision based on the threshold. For each document retrieved, we will see whether it is relevant and do some adaptation accordingly. The adaptations including adjusting the weight of winnow classifier and modifying the feature vectors and threshold. The threshold-adjusting algorithm is the heuristic algorithm we presented at TREC10 [Wu01]. The winnow's weight adjusting is the same as described in 1.1.1.

1.3 Effect of Winnow classifier and analysis

To see the effect of winnow classifier, we investigate the words in both winnow classifier and the vectors gotten from training set and find that only about 36% of the words of winnow classifier has occurred in the vectors. Therefore, there must exist many documents on that winnow classifier and VSM will make different decisions. If these documents happen to be irrelevant, combining winnow classifier with normal VSM will enhance the system's precision. We have done several experiment and found that Winnow classifier's recall is very high (0.5137 in training set and 0.4360 in testing set) but it's precision is very low (0.1406 in training set and 0.0823 in testing set). So combining winnow classifier with VSM will lower the system's recall, but will enhance the system's precision at the same time. Since the system's performance is the function of precision and recall and emphasize on precision, combine normal VSM with winnow classifier will enhance system's performance ultimately. The experiment results have confirmed this assumption. Table 1.1 is the experiment results of filtering system using winnow classifier and Table 1.2 without winnow classifier. We can see the efficiency of using winnow classifier.

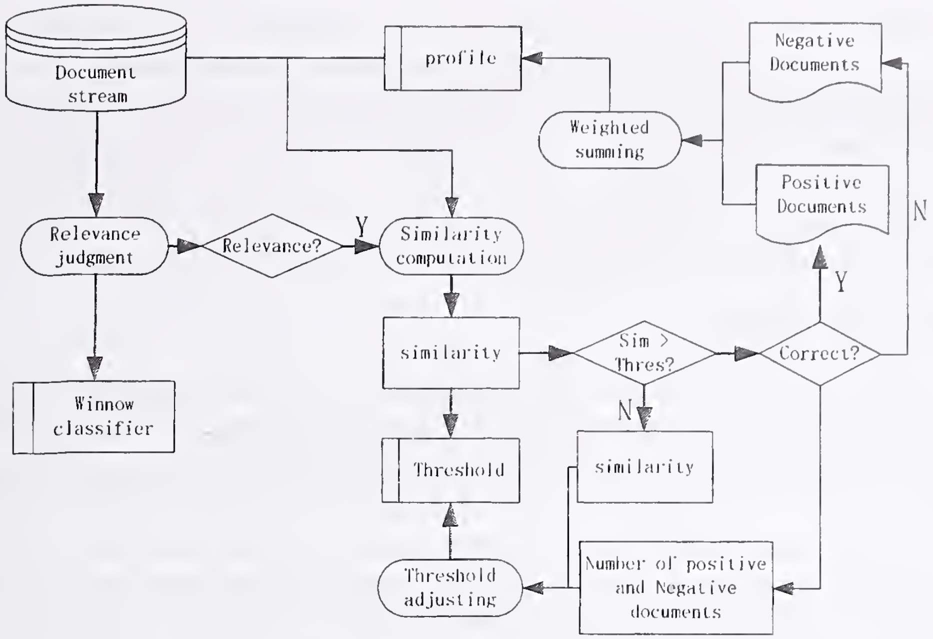


Figure 1.2 Architecture of adaptation stage

Table 1.1 Experiment results of filtering system based on normal VSM

VSM	Recall	Precision	T11F	T11SU
Total	0.2026	0.1544	0.1271	0.1841
R101 ~ R150	0.3159	0.2668	0.2204	0.2287
R151 ~ R200	0.0894	0.0421	0.0337	0.1394

Table 1.2 Experiment results of filtering system combine normal VSM with winnow classifier

Winnow	Recall	Precision	T11F	T11SU
Total	0.1390	0.2456	0.1771	0.2334
R101 ~ R150	0.2240	0.4511	0.3189	0.3657
R151 ~ R200	0.0541	0.0401	0.0354	0.1011

We also conducted experiments to see the effect of dynamic adjusting winnow's weights during adaptive filtering. Table 1.3 and Table 1.4 show the efficiency of adjusting winnow's weights. We also have try the method that add the words of winnow classifier to topic vectors and try different way to assign weight to words. but the results are not satisfactory.

Table 1.3 Experiment results of filtering without adjusting the winnow weight

static winnow	Recall	Precision	T11F	T11SU
Total	0.1387	0.2471	0.1741	0.2484
R101 ~ R150	0.2257	0.4461	0.3064	0.3556
R151 ~ R200	0.0517	0.0481	0.0417	0.1412

Table 1.4 Experiment results of adjusting the winnow weight while filtering

Dynamic winnow	Recall	Precision	T11F	T11SU
total	0.1382	0.2573	0.1788	0.2735
R101 ~ R150	0.2278	0.4595	0.3116	0.3717
R151 ~ R200	0.0485	0.0550	0.0460	0.1754

2. Question Answering

Since the TREC evaluation for Question Answering begins four years ago, quite a few sites joined in this task. The deep NLP method get succeed during the first three years [Harabagiu99], while the shallow method [Soubbotin01] get even greater success in last year's evaluation. Their success propels us to focus on the shallow NLP method.

Our QA system can be divided into four modules: pre-processing and indexing (the offline model), question analysis, sentence searching, and answer finding. Moreover, the last module can be divided into two sub-modules: answer extracting & meriting, and confidence calculation & sorting. Among them, we pay much attention on the second and last modules, which can be shown in the next figure.

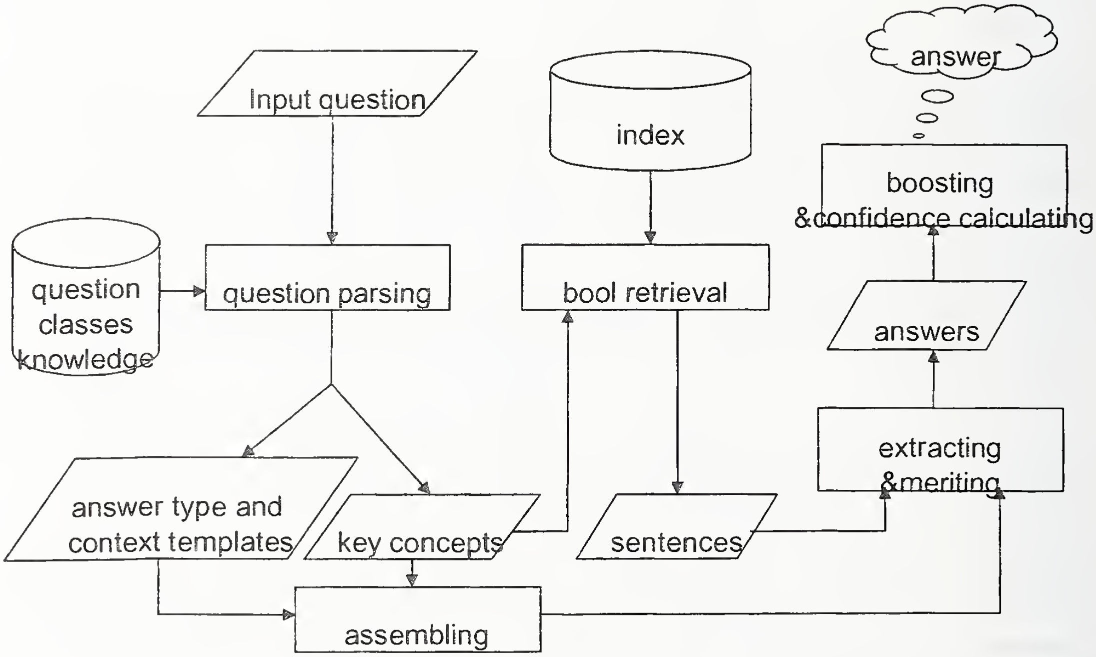


Figure 2.1 QA online modoles

From left to right: question analysis, sentence searching, answer finding

2.1 Building the QA knowledge base

The QA task is open domain. Quite a few participants use the open domain knowledge base such as WordNet to build their systems, while the QA task needs additional knowledge. How can the system deduce from the WordNet that, the answer of the question "*Who invented telephone*?" is a person name? There should be some knowledge base to tell the system that such kind of question's answer should be person name. Thus, almost every system has integrated more or less special knowledge for the QA task.

Our knowledge base includes about 80 question classes, which can be used in the Question Analysis model and Answer Extracting model. Each class includes three parts: question patterns, answer types and context templates. Following is an example:

```
<QuestionClass ID=302>
<Pattern ID=1> Who; <KeyConcept ID=1 Type=VBD>; <KeyConcept ID=2 Type=NP> ;QUERY; </Pattern>
<Pattern ID=2> Who;VBZ11;NP1; QUERY;</Pattern>
<Answer ID=1 Weight=1>    PersonName </Answer>
<Context ID=1 Weight=2> A;VBD1;NP2; </Context >
<Context ID=2 Weight=1.5> NP2;VBN1;by;A; </Context >
<Context ID=3 Weight=1.8> A;Comma;NP2;POS;NN1; </Context >
<Context ID=4 Weight=1.8> A;VERB_BE;NP1;of;NP2; </Context >
<Context ID=5 Weight=1> NP2;VERB_BE;VBN1;by;A; </Context >
<Context ID=7 Weight=1> A;POS;VBG1;IN;NP2; </Context >
<Context ID=8 Weight=1> A;IN;VBG1;NP2;</Context >
<AnyOrder ID=1 Weight=0.3> A;VBD1;NP2 </ AnyOrder>
</QuestionClass>
```

One question class may include one or more patterns, and these patterns determine whether a question belongs to this class or not. The question pattern may include normal words and key concepts. Every key concept is combined with ID and Type, which identify and recognize the concept.

The answer types are common, such as base NP, person name, location, time, etc.

The context template is quite like those of [Soubbotin01], but we use concept matching instead of its pure string matching method. We also divide the template into two types of strict order and lenient order.

2.2 Question Analysis

The pre-processing and indexing model is general. We analysis the morphology and delete the stop words based on the POS tagging. Only the nouns, verbs, adjectives, and adverbs are indexed.

We divide the key word into two categories: words that must appear and optional words, which are abbreviated as MA-Keywords and OA-Keywords. The OA-Keywords are replaceable words such as verbs. Others are all MA-Keywords.

When a question is presented, its question class is examined. If one question class's question pattern can match the question, the system classifies the question into this question class. Then the question will be divided into several parts by the question pattern. The system only regard the words belong to key concepts as the candidate keywords, while the others will be discarded as stop words. We then use some structures to discard other stop words, such as "the A of B" and if A is the hypernym of B, then A is the stop word. For example, the phrase "the state of Alaska", only "Alaska" will be tagged as keywords. Then we discard the words except noun, verb, adjective and adverb. Finally, we divide the remaining words into MA-keywords and OA-Keywords by the above principle. Then the system acquires the corresponding context patterns and answer types.

2.3 Answer extraction and merit

There is a change in this year's task. Instead of five 50 bytes (or 250 bytes) answer gobbets, every question should return an exact answer. This change requires us paying more attention to answer itself.

In order to get a high precision and a moderate recall, we limit the answer from both the inside and outside pattern of the answer. We divide the context template and answer type into two categories: the strict and the lenient. When assembling the context template and answer type, every combination is allowed, except both lenient.

Each term in the context template, including the answer, is a concept. System first locates all the concepts in the coming sentence, then checks if the sentence matches the pattern, and extract the answer if success.

One answer's score equals the summation of the score of context template, answer type, and sentence match score and concepts match score.

$$S_{ans}=S_{ct} + S_{at} + S_{sen}+ \sum S_c$$

Where, S_{ans} is the score of the candidate answer, S_{ct} is the score of context template, S_{at} is the score of answer type, and S_c is score of the concepts.

Concept match is based on keyword matching:

$$S_c =\sum S_{mk} / \text{total keywords in the concepts of the question}$$

$$S_{sen}=\sum S_{mk} / \text{total keywords in the question}$$

Where, S_{mk} is the score of matched keywords. If two keywords are matched by original or derivation form, S_{mk} is set to 1.0. If keyword in the question is the hypernym of the keyword in the sentence, , S_{mk} is set to 0.8. Otherwise, the concept cannot be matched.

2.4 Confidence

As known to all, another dramatic change in this year's task is that the submitted questions should be ranked by confidence. This means that we should be aware of what we have submitted. We use the ME model to determine every answer's confidence[Berger96].

According to our maximum entropy based sorting method, the rank of different answer is determined by several weighted factors, and the confidence score is the product of the exponent of the weights of every factors. The weight of every factor is assigned during the training of TREC-10 questions.

As for the non-nil answers, five factors are considered, which are:

- Score (The score of S_{ans} is dependent on the question, and cannot be used to compare the confidence of different questions directly.)
- Step (The answer can be extracted in step1 or step2. Step1 use strict answer patterns, while step2 use lenient patterns, so step1 is better.)
- BestCount (For each question, a lot of answer snippet can be found, some of which are the same. We use boosting algorithm to find the best answer, and the best answer can be find in several places, the number of which is the BestCount.)
- BestCount/totalCount (totalCount is the total of possible answer snippets.)
- Answer Type (Person, Place, Time, etc.)

As for the nil answers, five factors are considered, which are:

- The number of key words (Key words are those words which are important to answer finding in the question. They are extracted in our question analysis module.)
- The number of key words which cannot be matched between questions and TREC corpus.
- The total number of question words (key words as well as optional words)
- The number of all the question words, which cannot be matched between questions and TREC corpus.
- Answer Type

Following are some of our experiment results, where the training data are the 500 questions of TREC-10, and the test data are the 500 questions of TREC-11.

Sorted by question id: 0.315 (this is the baseline)

Four factors (open test, answer type is not considered): 0.461

Five factors (open test): 0.434

Four factors (closed test): 0.498

Five factors (closed test): 0.489

Therefore, our maximum entropy method does significantly better than the baseline method. In addition, only consider the first four factors is better.

3. Web Retrieval

This is year is the second year that we attend TREC Web task. We have submitted five runs for the topic distillation task: fduwt11b0, fduwt11t1, fduwt11o1, fduwt11t2, fduwt11o2. Detailed information of each run is given in the following table.

Table 3.1 Web runs submitted

Run ID	Information used
fduwt11b0.submit	Use basic search
fduwt11o1.submit	Use title and link structure to expand and re-sort the basic search result
fduwt11t2.submit	Use title and anchor text to expand and re-sort the basic search result
fduwt11t1.submit	Use title and link structure to expand and re-sort the basic search result
fduwt11o2.submit	Use title, anchor text and link structure to expand and re-sort the basic search result

3.1 System Architecture

The Topic Distillation task of this year requires the retrieval system return “key resources” in the .GOV corpus for certain queries. Compared with last year’s relevance retrieval task, the amount of “key resources” the system should return is not as many as relevant documents in older task. However, the quality of the key resources is more important than that of older task. Considering this specialty, we modified our original search system to let it return high precision result. The main idea does not change a lot; we use the basic search algorithm to get a set of relevant documents. Then special post processing modules are used to expand and re-sort the base set using title, link structure and anchor text information. The final result set is a set of desired key resources.

The process of indexing includes:

- Transform HTML files in the corpus into plain text. At the same time of transformation, we use special module to extract information of links with anchor text for constructing link database later.

- Index the plain text corpus for basic search algorithm
- Index the link database

The process of searching includes:

- Make morphology analysis of queries.
- Use basic search algorithm to get base set of relevant documents.
- Use post-processing modules to do expansion and re-sorting upon the base set, get the final result of "key resources".

3.2 Improved kernel search algorithm

This year we have made some modifications on the kernel search algorithm based on "shortest extend". The goal is to improve the precision of the first retried documents. We believe this will make it easy for the post-processing module to expand and re-sort.

We modified the score equation of calculating the shortest extend. The purpose is trying to avoid the situation that extents of queries longer than two words will get extremely low scores.

For shortest extent (p, q). its the score I(p, q) is

$$I(p,q) = \begin{cases} \left(\dfrac{K}{q-p+1}\right)^{a} & 若\ q-p+1 >= K \\ 1 & 若\ q-p+1 <= K \end{cases} \tag{3.1}$$

in which, we set 'a' to1, K=16 * (length(Q) – 1), length(Q) represents the number of the word in the query Q.

We noticed that the average length of document in the .GOV corpus is longer than that in older corpus. Thus, we altered the method of getting document score. The basic idea is to lower the score of the document if the document length exceeds the average document length. The equation is show as below:

$$S(D) = \sum_{(pi,qi) \in D} I(p_i,q_i) * w1(p_i,q_i)$$

$$\text{if (length(D)} > \text{AVG\_DOC\_LEN)} \tag{3.2}$$

$$\text{S(D)} = \text{S(D)} * \frac{\text{AVG\_DOC\_LEN}}{\text{length(D)}}$$

in which, length(D) represents the number of the word in document D. AVG_DOC_LEN represents the average length of documents in the corpus, also counted in word number.

3.3 Post process module

We use two ways to expand and re-sort the base set of relevant documents. Details are given in the following.

1) Use structured information to improve expand base set

Words in different parts of a document have different importance. For example, Words in title are more important than words in content; words of large font are more important than words of small font; words of bold or underlined font are more important than words of normal font. In this year's web track, we used title information to improve our retrieval result.

We increase the score of a document if the title of this document includes retrieval words. The increment coefficient is between 0 and 1:

$$\text{score\_title}_d=\left(\frac{\text{count\_of\_match}}{\text{count\_of\_retrieval\_words}}\right)^{\gamma} \qquad (3.3)$$

in which, γ is a parameter greater than 1.

2) Use breadth-first traverse algorithm to find out key resource

By breadth-first traverse algorithm, we access all pages of a domain from the homepage of the domain. Then we construct a tree, the root of which is the homepage of the domain.

Our retrieval procedure has two steps:

Step 1: Retrieve a batch of documents using our text-only search engine. Every document retrieved is given a score. This score of document d is called score_ text$_d$.

Step 2: Calculate key resource score of a document score_KR

$$\text{score\_KR}_i=\sum_{j\in \text{subtree}(i)} \frac{\text{score\_text}_j}{\gamma^{\text{level}(i,j)}} \qquad (3.4)$$

in whick, γ is parameter greater than 1.

3) Synthesize score_ text, score_title, score_KR

final_score$_d$= α * score_ text$_d$+ β * score_title$_d$+ γ * score_KR$_d$

then we re-sort documents on final_score, rettun the top n documents as our result.

Because the task of "Topic Distillation" is a totally new task, we cannot use data in the past task for training. We have made a human tagging system to let several students tagging our experiments result at the same time. In this way, we can get some feedback from human.

Experiment results show that the above algorithm can lead to satisfactory results. The average P@10, P@20, P@30 of 49 queries are 0.45, 0.31, 0.26 respectively, while the average median precision is 0.11. 0.09, 0.08.

4. Video Track

On Video Track of this year, we paticipated in Shot Segmentation, Feature Extraction and Search task.

4.1 Shot Segmentation

This year we use most parts of TREC-10 Shot Segmentation System [Wu01]. FFD (*Frame-to-Frame Difference*) calculated by Luminance Difference and Color Histogram Similarity are used to detect the Shot Changes. We use two thresholds θ_C and θ_G, which are calculated automatically according to the FFD value histogram in 500 frames, to detect if there is a clear FFD value change caused by Shot Changes. Then Flashlight Detection and Motion Detection are applied for candidate Shot Changes to remove the false alarms of Cut and Gradual. The parameters used in the system are trained and adjusted based on the TREC-10 Video Library. According to the performance on TREC-10 Video Library, we selected the system parameters to generate the submissions.

We add Fade In/Out Detection into our system this year although Shot Segmentation task does not include it. In our submissions, Run02, Run09 and Run10 include Fade Detection. In our system, Fade In/Out Detection is applied to all candidate Gradual Changes. If a black screen chain exists in the candidate duration, we think it is a Fade. Otherwise, it will be labeled as Dissolve. In order to detect the black screen and get the accurate boundary of Fade, a frame is split into several blocks whose size is 8×8 (pixels). *Maximum Luminance* $Lum_{max}(n)$ and *Black Value* $Black(n)$ are calculated for each frame *n* in the candidate duration.

$$Lum_{max}(n) = \sum_{block_k \in B_{max}} Luminance(n,k) \tag{4.1}$$

$$B_{max} = \{block_{i1}, \cdots, block_{i10}\}\ are\ set\ of\ ten\ blocks\ with\ maximum\ Luminance\ value$$

$$Lum_{min}(n) = \sum_{block_k \in B_{min}} Luminance(n,k) \tag{4.2}$$

$$B_{min} = \{block_{i1}, \cdots, block_{i10}\}\ are\ set\ of\ ten\ blocks\ with\ minimum\ Luminance\ value$$

$$Black(n) = \frac{Lum_{max}(n)}{10} \times \frac{Lum_{max}(n)}{Lum_{min}(n)+1} \tag{4.3}$$

Then system finds four points in the candidate duration:

- Black Screen Start Point: the first frame *n* satisfies $Lum_{max}(n) < th_{black}$.
- Black Screen End Point: the first frame *n* satisfies $Lum_{max}(n) \geq th_{black}$.
- Black Decrease Point: the frame *n* which has the maximum Black Value between Gradual Start Frame and Black Screen Start Point.
- Black Increase Point: the frame n which has the maximum Black Value between Black Screen End Point and Gradual End Frame.

If Black Screen Start Point is equal to the Gradual Start Frame, we regard it Fade In. It starts from Black Screen Start Point and ends at Black Increase Point. On the contrary, if Black Screen End Point is equal to the Gradual End Frame, a Fade Out, which starts from Black Decrease Point and ends at Gradual End Frame, is detected. The third condition is that the black screen chain located in the middle of candidate duration. At this time, a Fade Out followed by a Fade In will be detected. It starts at Black Decrease Point and ends at Black Increase Point.

Evaluation shows that our system has a good balance between precision and recall. Comparing F-Value, the rank of our best result for all the changes, Cut Changes and Gradual Changes is 3, 3 and 9 (out of 54 systems). On Gradual Accuracy, frame-recall of our system is better than frame-precision. Comparing with other submitted systems, our system located at the middle on Gradual Accuracy.

4.2 Feature Extraction

According to the FE Task of this year, we developed a new Video Feature Extraction System. It consists of five sub-systems: Outdoor / Indoor Detection, Cityscape / Landscape Detection, Face / People Detection, Text Detection and Speech / Music / Monologue Detection. In each sub-system, a value calculated by whatever methods and features is used for ranking. In the following part, we call it "Ranking Value". Evaluation shows that our system works well on these features: Cityscape, Landscape, Indoor and Music.

4.2.1 Outdoor / Indoor Detection

In our Outdoor / Indoor System, we use K-Nearest Neighbor Classifier. Considering the difference between Outdoor and Indoor, we select Color Histogram and Edge Direction Histogram as the feature [Szummer98]. A 512-bin color histogram is calculated in Color Space T=R-B. The Edge Direction is obtained by calculating the ratio of the grades on vertical direction and horizontal direction at every edge point. The edge point is got by Canny Edge Detector. In the Edge Direction Histogram, all the directions are separated as 12 bins.

The training set includes 600 indoor images and 1300 outdoor images. They are selected from Feature Dev Set. For each shot in Feature Test Set, we only apply the classifier to the selected keyframe for a shot. In Run01, the keyframe are divided into 4×4 blocks. The histogram distance is calculated for each block. After summing with weights, we will get the distance for whole keyframe. On the contrary, we only calculate the distance on whole keyframe in Run02. In each run, the Ranking Value is minimum distance between keyframe and the training images.

4.2.2 Cityscape / Landscape Detection

Similar with Outdoor / Indoor System, we also use K-Nearest Neighbor classifier in Cityscape / Landscape Classification. However, we only select Edge Direction Histogram as the feature in this system [Vailaya98].

The training set includes 410 cityscape images and 250 landscape images. They are also selected from Feature Dev Set. We think all the cityscape and landscape image should be outdoor image at first. Therefore, the outdoor Ranking Value calculated in Outdoor / Indoor Detection will multiply with the maximum distance output by Cityscape / Landscape Classifier. The product is the Ranking Value in this system. In the submissions, Run01 use the Ranking Value of Outdoor/Indoor Detection Run01 and Run02 use the Ranking Value of Outdoor/Indoor Detection Run02.

4.2.3 Face / People Detection

This system uses the same idea of TREC-10. The method consists of three steps: Skin-Color based Segmentation, Motion Segmentation, and Shape Filtering. Considering the difference between the TREC-10 Video and TREC-11 Video, some new training data selected from Feature Dev Set are added into the Skin-Color template.

We use the following equation to calculate the Ranking Value for each shot:

$$RankingValue_{face} = \frac{\#\,of\ frames\ contain\ face}{\#\,of\ frames\ in\ a\ shot} \tag{4.4}$$

$$RankingValue_{people} = \frac{\#\,of\ frames\ contain\ two\ or\ more\ faces}{\#\,of\ frames\ in\ a\ shot} \tag{4.5}$$

4.2.4 Text Detection

Same with last year's work, there are still three main parts in our Video Text Detection System: Text Block Detection, Text Enhancement and Binarization. In order to reduce the false alarms, we combine Neural

Network [Li00] as a preprocessing in our Text Block Detection.

System applies 2-level Harr Wavelet Decomposition on the image. The image will be split as four sub-bands at each level: HH, HL, LH and LL. For each N×N window, we can calculate three features:

$$E(I) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} I(i,j) \qquad (4.6)$$

$$\mu_2(I) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} (I(i,j) - E(I))^2 \qquad (4.7)$$

$$\mu_3(I) = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} (I(i,j) - E(I))^3 \qquad (4.8)$$

Totally, we can get 24 (2 [level] ×4 [sub-band] × 3 [feature] = 24) features. Eight features are selected as the input of the Neural Network.

The output of Neural Network is a confidence between 0 and 1. The more the confidence approaches 1, the more possible the window is classified as text. A threshold is used to determine whether it is a text or non-text. We select three-layer BP Neural Network as a classifier to identify text regions. Bootstrap method is used in training. The training data come from Feature Dev Set and TREC-10 Video Library.

For each shot, the system processes only one frame in every ten. On each processed frame, we use a small window (16×16 pixels) to scan the image and classify each window as text or non-text using trained neural network. When scanning the image, we move the window 4 pixels at a time. If a window is classified as text, all the pixels in this window are labeled as text. Those pixels which are not covered by any text window are labeled as non-text. Then we generate a binary map from original image. The following Text Block Detection is only applied in the region labeled as text. The experiments show that the text detection precision increases about 30% while the recall almost does not decrease.

We average the output values generated by Neural Network for all the small windows in one frame. This average value is the confidence that a single frame contain text region. We select the maximum frame confidence of all the frames in a shot as the ranking value.

4.2.5 Speech / Music / Monologue Detection

Speech/Music Classification is applied on the 1-second window. The features we used include: Mean and Covariance of Zero-crossing Rate, High Zero-crossing Rate Ratio, Mean and Covariance of Short Time Energy, Low Short Time Energy Ratio, Noise Frame Ratio, Mean and Covariance of Brightness, Spectral Flux, Spectral Roll-off Point, Mean and Covariance of LPC, Mean and Covariance of MFCC, Mean and Covariance of Pitch, Mean and Covariance of Band Spectrum, Mean and Covariance of Band Width [Lu01][Scheirer97].

Nearest Neighbor Model and Gaussian Mixture Model are trained by TREC-10 Videos. Applying these trained models on 1-second window, we can get the type of each window. In our submission, Run01 uses NN Model and Run02 uses 16-mixture GMM Model.

The Ranking Value of speech, music and monologue are calculated by:

$$Ranking\ Value_{speech} = \frac{\#of\ \ windows\ whose\ type\ is\ speech}{\#of\ \ windows\ in\ a\ shot} \qquad (4.9)$$

$$Ranking\ Value_{music} = \frac{\#of\ \ windows\ whose\ type\ is\ music}{\#of\ \ windows\ in\ a\ shot} \qquad (4.10)$$

$$Ranking\ Value_{monologue} = RankingValue_{speech} \times RankingValue_{face} \qquad (4.11)$$

4.3 Search

We have submitted four runs in Search Task. Considering the difficulty of search topics, not all of the topics are processed in each run.

The whole architecture of the Search system is almost same with last year [Wu01]. However, there are some improvements in Face Recognition and Object Search.

4.3.1 Face Recognition

In TREC-11 Search Task, there are four topics concerning about a certain people. Face Recognition is the basis for such topics.

The eigen-face and Fisher method are commonly used in face recognition especially when the person set is completely known. However, in most cases of video retrieval the complete person set cannot be acquired. This year we tried a fast manifold-based approach to face recognition in TREC-11 Search Task. It can be used when there are only few different images of a specific person and runs fast.

Let $X_1, X_2, \cdots, X_n \in R^d$ be the vectors of images of a specific person. Usually n is small and the dimension d is very big, i.e. $n << d$

Denote $\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i, Y_i = X_i - \overline{X}, 1 \le i \le n$ and S the subspace spanned by $\{Y_i\}$. The set $M = \{X : X = \overline{X} + Y, Y \in S\}$ is called manifold.

Denote $S^{\perp}$ the orthogonal complement subspace of *S*. It is easy to see that if $X \in M, (X - \overline{X}) \in S$ i.e. $P_{S^{\perp}}(X - \overline{X}) = \Phi$, or $\left\| P_{S^{\perp}}(X - \overline{X}) \right\| = 0$, where $P_S(X)$ is the projection of *X* on *S*.

Based on observation above, we scan the input image with sub-windows of different positions and levels of the image pyramid. Then the distance between the scanning window and the mean of samples in $S^{\perp}$ is calculated. The sub-window with the minimum distance may include the person we search for if the minimum distance is below a threshold.

Experiment shows this approach works quite well even in case where there are only few samples. However, it is quite time-consuming.

To speed up, we collect a large number of non-face samples $U_1, U_2, \cdots, U_m$, where m is very big. Denote $V_i = P_{S^{\perp}}(U_i - \overline{X}), 1 \le i \le m$. Note that if *X* is a face then $P_{S^{\perp}}(X - \overline{X}) \approx \Phi$ and the dimension of

$S^{\perp}$ is *d-n* and very big.

We do the principle component analysis with $V_1, V_2, \cdots, V_m$ and get the first few principle components $Z_1, Z_2, \cdots, Z_k$, where *k* is small. Denote *Z* the subspace spanned by $Z_1, Z_2, \cdots, Z_k$.

Then based on the note above, we can filter out the input image *X*, if $\left\| P_Z\left(X - \overline{X}\right) \right\| > th$. Since *k* is much smaller than *d-n*, this filtering step is much faster. In our experiment, applying this step will make the face recognition 5-fold faster and with almost no decreasing of performance.

4.3.2 Color Histogram Comparison

Color Histogram similarity is used to compare the Image Example and Key Frame of each shot. It will provide us the similarity on the image between Image Example and Key Frame. We calculate the histogram in RGB and YUV space. During the calculation and comparison, two modes are used:

- Whole Image Mode: For both Key Frame and Image Example, the histogram is calculated on the whole image.
- Block mode: For Key Frame, we split it into several blocks with different size. The histogram is calculated on each block. For Image Example, the histogram is calculated on the whole image. Then the histogram comparison is processed between the histogram of each block and image example. The maximum similarity will be selected as the final similarity.

In searching, Block mode is used on the topics which is concerning about a certain object. Such as parrots, butterfly and so on.

4.3.3 Searching

For each topic, we combine the similarities come from different modules. Such as Face Recognition, Text Recognition, Color Histogram Comparison, ASR Text etc.

In our submission, Sys1 only use the information get by our own search modules. There is no ASR Text and Feature Extraction results are used. However, Feature Extraction Confidence is useful for some topics. For example, Face Confidence is useful to certain people searching and Cityscape Confidence is useful for Topic 86. So in Sys2 and Sys3, we combined feature extraction confidence into the searching. Sys2 use our own Feature Extraction results and Sys3 use the reference Feature Extraction results provided by IBM and MediaMill. In Sys4, we combine the ASR Results provided by LIMSI. We select some keywords manually for each topic. These selected keywords are used for Text Retrieval on ASR Results.

NIST's evaluation shows that our searching system is not efficient in several topics. In the future work, we should pay more attention on Image Similarity calculation.

ACKNOWLEDGMENTS

This research was partly supported by NSF of China under contracts of 69935010 and 60103014, as well as the 863 National High-tech Promotion Project of China under contracts of 2001AA114120 and

2002AA142090. We are thankful to Lin Mei, Yuefei Guo, Sitong Huo, Yi Zheng, Xin Li, Kaijiang Chen, Xiaoye Lu, Jie Xi, He Ren, Li Lian, Wei Qian, Hua Wan, Tian Hu, Jiayin Ge, Jian Gu, Danlan Zhou, Zhiyan Tang, Xipeng Qiu, Lan You, Lin Zhao, Rongrong Wang, Min Jin, Jiawei Rong, Wanjun Jin for their help in the implementation.

Reference

[Berger96] Adam L. Berger, Stephen A. Della Pietra, Vincent J. Della Pietra, *A Maximum Entropy Approach to Natural Language Processing*, ACL'96

[Harabagiu99] Sanda Harabagiu and Dan Moldovan. *FALCON: Boosting Knowledge for Answer Engines.* Proceeding of The Eighth Text Retrieval Conference, November, 1999

[Li00] Huiping Li, *Automatic Processing and Analysis of Text in Digital Video*, Technical Report of Center for Automation Research, University of Maryland College Park, December 2000

[Littlestone88] N.Littlestone, *Learning quickly when irrelevant attributes abound: a new linear threshold algorithm*, Machine Learning, 2: 285-318, 1988

[Lu01] Lie Lu, Hao Jiang and Hongjiang Zhang, *A Robust Audio Classification and Segmentation Method*, Proc. of the 9th ACM International Multimedia Conference and Exhibition, pp. 103-211, 2001

[Scheirer97] E. Scheirer and M. Slaney. *Construction and Evaluation of a Robust Multifeature Music/Speech Discriminator*. Proc. of ICASSP'97, vol. II, pp 1331-1334. IEEE, April 1997

[Soubbotin01] M. M. Soubbotin, S.M. Soubbotin. *Patterns of Potential Answer Expressions as Clues to the Right Answers.* Proceeding of The tenth Text Retrieval Conference, November, 2001

[Szummer98] Martin Szummer, *Indoor-Outdoor Image Classification*, IEEE International Workshkop on Content-based Access of Image and Video Databases, in conjunction with ICCV'98, Jan. 1998

[Vailaya98] Aditya Vailaya, Anil Jain, HongJiang Zhang, *On Image Classification: City Images vs. Landscapes*, Pattern Recognition, Vol.31, pp1921-1936, Dec. 1998

[Wu01] Lide Wu et al., *FDU at TREC-10: Filtering, Q&A, Web and Video Tasks*. Proceeding of The tenth Text Retrieval Conference, November, 2001

Experiments in Named Page Finding and Arabic Retrieval with Hummingbird SearchServer$^{\text{TM}}$ at TREC 2002

Stephen Tomlinson
Hummingbird
Ottawa, Ontario, Canada
stephen.tomlinson@hummingbird.com
http://www.hummingbird.com/

February 7, 2003

Abstract

Hummingbird participated in the named page finding task of the TREC 2002 Web Track (find the named page in 18GB from the .GOV domain) and the monolingual Arabic topic relevance task of the TREC 2002 Cross-Language Track (find all relevant documents in 869MB of Arabic news data). In the named page finding task, SearchServer returned the named page in the first 10 rows for more than 80% of the 150 queries. Searching the full document content produced mean reciprocal rank (MRR) scores more than 20 points higher than just searching particular HTML properties (such as the Title), but enhancing a content search with a little extra weight for HTML properties further increased MRR by 6 points (with standard error of just 2 points). Treating queries as phrases was not found to help significantly (on average), but document length normalization increased MRR by more than 20 points. For Arabic topic relevance, light algorithmic stemming increased mean average precision (MAP) by 5 points, use of Arabic stop words increased MAP by 1 point, and query expansion from blind feedback increased MAP by 3 points.

1 Introduction

Hummingbird SearchServer[1] is an indexing, search and retrieval engine for embedding in Windows and UNIX information applications. SearchServer, originally a product of Fulcrum Technologies, was acquired by Hummingbird in 1999. Founded in 1983 in Ottawa, Canada, Fulcrum produced the first commercial application program interface (API) for writing information retrieval applications, Fulcrum® Ful/Text$^{\text{TM}}$. The SearchServer kernel is embedded in many Hummingbird products, including SearchServer, an application toolkit used for knowledge-intensive applications that require fast access to unstructured information.

SearchServer supports a variation of the Structured Query Language (SQL), SearchSQL$^{\text{TM}}$, which has extensions for text retrieval. SearchServer conforms to subsets of the Open Database Connectivity (ODBC) interface for C programming language applications and the Java Database Connectivity (JDBC) interface for Java applications. Almost 200 document formats are supported, such as Word, WordPerfect, Excel, PowerPoint, PDF and HTML.

SearchServer works in Unicode internally [5] and supports most of the world's major character sets and languages. The major conferences in text retrieval evaluation (TREC [10], CLEF [2] and NTCIR [7]) have provided opportunities to objectively evaluate SearchServer's support for a dozen languages.

[1] Fulcrum® is a registered trademark, and SearchServer$^{\text{TM}}$, SearchSQL$^{\text{TM}}$, Intuitive Searching$^{\text{TM}}$ and Ful/Text$^{\text{TM}}$ are trademarks of Hummingbird Ltd. All other copyrights, trademarks and tradenames are the property of their respective owners.

This paper looks at experimental work with SearchServer for named page finding (just one right answer, i.e. a known-item search task) and monolingual Arabic retrieval (find all the relevant documents, i.e. a topic relevance task). All experiments were conducted on a single-cpu desktop system, OTWEBTREC, with a 600MHz Pentium III cpu, 512MB RAM, 186GB of external disk space on one e: partition, running Windows NT 4.0 Service Pack 6. For the submitted runs in July 2002, an internal development build of SearchServer 5.3 was used (5.3.500.264).

2 Named Page Finding

The .GOV collection of the TREC 2002 Web Track consists of pages downloaded from the .gov domain of the World Wide Web in early 2002. It was distributed on 7 CDs. We copied the contents of each CD onto a "compressed NTFS" area of OTWEBTREC's e: drive (e:\data\compressed\gov\cd1 - e:\data\compressed\gov\cd7). The TRANS.TBL files were not considered part of the collection and were removed. The 4613 .gz files comprising the collection were uncompressed (and Windows NT internally recompressed them on the compressed NTFS drive). Uncompressed, the 4613 files consist of 19,455,030,550 bytes (18.1 GB). Based on the change in bytes free on the drive, the files occupied 9,329,721,344 bytes (8.7 GB) on the compressed NTFS drive. Hence, NTFS compression saved about 9.4 GB of space, noticeably less than gzip compression (which saved 13.9 GB). Each file contains on average about 270 "documents", for a total of 1,247,753 documents. The average document size is 15,592 bytes. For more information on the .GOV collection, see [4].

2.1 Indexing

The custom text reader called cTREC, described in last year's paper [13], was enhanced for the named page finding experiments.

In expansion mode, cTREC, which previously just extracted the DOCNO identifier, was enhanced to support a /H option for extracting the following property information from each .GOV document's header and content for storage in columns 129-140 of the SearchServer table:

- 129: "non-empty" title, which was filled with the first non-empty value of columns 130-135 (i.e. the title was used if there was one, otherwise the meta title was used if there was one, etc., with the URL used as a last resort).
- 130: title (text following <TITLE> up to </TITLE>, if any).
- 131: meta title (content of the META TITLE tag, if any).
- 132: meta subject (content of the META SUBJECT tag, if any).
- 133: meta description (content of the META DESCRIPTION tag, if any).
- 134: first heading (text following the first occurrence of the <H1>, <H2> or <H3> tag up to its closing tag, if any).
- 135: URL (which was always included in the DOCHDR section, before the document content).
- 136: URL type (calculated from the URL as described below).
- 137: URL depth (calculated from the URL as described below).
- 138: meta keywords (content of the META KEYWORDS tag, if any).
- 139: all properties except keywords (i.e. a concatenation of columns 130-135).
- 140: all properties (a concatenation of columns 139 and 138).

The URL was truncated at 256 bytes (only 5 were longer), and the other properties were truncated at 1024 bytes.

The URL type was set to ROOT, SUBROOT, PATH or FILE, based on the convention which worked well last year for the Twente/TNO group [14] on the entry page finding task (also known as the home page finding task). Our exact rules were as follows. The slash count of the URL was calculated (a count of the '/' characters not including the leading "http://"). The URL was considered of homepage-type if it ended with "/", "/index.html", "/index.htm", "/default.html", "/default.htm", "/default.asp", "/home.html", "/home.htm", "/welcome.html" or "/welcome.htm" (case-insensitive comparisions were used). ROOT was assigned if the URL was of slash count 0 or was a homepage-type URL of slash count 1. SUBROOT was assigned if the URL was a homepage-type URL of slash count 2. PATH was assigned if the URL was a homepage-type URL of slash count 3 or more. FILE was assigned for all other URLs.

The URL depth was based on the sum of the slash count and the node count, minus one if the URL was of homepage-type. The node count was the count of the dots before the first slash after "http://" (not counting the first dot if the URL began with "http://www.") plus the number of "?", ";" or "#" characters. (As every URL contained ".gov", the URL depth was guaranteed to be at least 1.) For convenience of wildcard searching and readability, the depth was converted to a term as follows: 1 was assigned URLDEPTHA, 2 was assigned URLDEPTHAB, 3 was assigned URLDEPTHABC, etc. with depths greater than 25 treated the same as 25.

In format translation mode, cTREC was enhanced to support a /q option which resumed indexing at the first quotation mark inside a tag, rather than always waiting for the end of the tag to resume indexing. This option hence would index potentially helpful text such as VALUE fields of INPUT tags and NAME fields of IMG tags, although more noise would also be indexed.

For the .GOV collection, the documents were assumed to be in Latin-1, and as for the web collections of past years, the /w option of cTREC was used to convert non-ASCII Latin-1 bytes to the ASCII range (if any occurred).

A SearchServer table called GOV was created for the .GOV collection with the following SearchSQL statement:

```
create schema GOV
create table GOV
(
DOCNO varchar(256) 128,
NONEMPTY_TITLE varchar(2048) 129,
TITLE varchar(2048) 130,
META_TITLE varchar(2048) 131,
META_SUBJECT varchar(2048) 132,
META_DESCRIPTION varchar(2048) 133,
FIRST_HEADING varchar(2048) 134,
URL varchar(2048) 135,
URL_TYPE varchar(2048) 136,
URL_DEPTH varchar(2048) 137,
META_KEYWORDS varchar(2048) 138,
ALL_BUT_KEYWORDS varchar(2048) 139,
ALL_PROPS varchar(2048) 140
)
periodic
stopfile 'mygov.stp'
basepath 'e:\data\compressed';
```

The DOCNO column was assigned number 128 and the remaining columns were assigned numbers 129-140 to correspond to the properties written by the /H option the cTREC text reader. (The reserved external text column, FT_TEXT, which corresponds to the document content, does not need to be specified in the schema.) The mygov.stp stopfile of 99 stop words is a little different from previous years in that it no longer contains single letters or any numbers.

Into the GOV table, just one row was inserted, specifying the top directory of the data set relative to the basepath:

```
insert into GOV ( ft_sfname, ft_flist )
values ( 'gov', 'cTREC/E/d=128/H:s!cTREC/w/q/@:s');
```

To index the GOV table, a Validate Index statement was executed:

```
validate index GOV validate table;
```

2.2 Searching

For the named page finding task of the Web Track, the 150 "topics" were in a file called "web-named_page_topics.1-150.txt". The topics were numbered NP1-NP150, and each contained a description of a page (e.g. "visiting pandas national zoo"). The task was to rank the named page as highly as possible. The topics were assumed to be in the Latin-1 character set, the default on North American Windows systems (though accent-sensitive searching was not enabled for the GOV table).

For the submitted hum02pd run of the named page finding task, below is an example SearchSQL query. This query would create a working table with the 2 columns named in the SELECT clause, a REL column containing the relevance value of the row for the query, and a DOCNO column containing the document's identifier. The ORDER BY clause specifies that the most relevant rows should be listed first. The statement "SET MAX_SEARCH_ROWS 50" was previously executed so that the working table would contain at most 50 rows:

```
SELECT RELEVANCE('V2:3') AS REL, DOCNO
FROM GOV
WHERE
 (ALL_PROPS CONTAINS 'visiting pandas national zoo' WEIGHT 1) OR
 (ALL_PROPS IS_ABOUT 'visiting pandas national zoo' WEIGHT 1) OR
 (FT_TEXT IS_ABOUT 'visiting pandas national zoo' WEIGHT 10)
ORDER BY REL DESC;
```

The ALL_PROPS column contained all the properties of column 140 (described earlier), e.g. the title and meta description but not most of the document content.

The CONTAINS predicate does phrase searching, so the listed terms would have to occur adjacently in the specified order (except stop words). "SET PHRASE_DISTANCE 4" was previously specified so that there could be up to 4 characters between adjacent terms (plus additional whitespace). By default, the CONTAINS predicate does exact searching (i.e. no stemming), though some normalizations (e.g. case normalization and canonical Unicode) are still done. The motivation for including the query as a phrase was that it seemed the query might often be in the title or other property information of the document (e.g. a query in mind was "Washington State Legislature" (which was not one of the 150 official queries)). The phrase searching was just given one-tenth the weight of content searching for relevance ranking purposes. Experiments on last year's entry page finding task suggested a small weight was helpful (on average) but a strong weight hurt results.

The IS_ABOUT predicate uses SearchServer's Intuitive Searching, described in last year's paper [13]. It by default uses English stemming and just requires one of the terms to match. It was used with WEIGHT 1 on the ALL_PROPS column to increase the ranking of documents with the query in the title or other property information. It was used with WEIGHT 10 on the FT_TEXT column (which represents the external document). Again, these weights were chosen based on what worked well on the previous year's entry page finding task.

For the submitted hum02upd and hum02up runs, a higher weight was given to URLs of particular type and depth, using a SearchSQL WHERE clause of the following form which was was found to work well on last year's entry page finding task:

```
WHERE
((ALL_PROPS CONTAINS 'visiting pandas national zoo' WEIGHT 1) OR
 (ALL_PROPS IS_ABOUT 'visiting pandas national zoo' WEIGHT 1) OR
 (FT_TEXT IS_ABOUT 'visiting pandas national zoo' WEIGHT 10)
) AND (
 (URL_TYPE CONTAINS 'ROOT' WEIGHT 10) OR
 (URL_TYPE CONTAINS 'SUBROOT' WEIGHT 10) OR
 (URL_TYPE CONTAINS 'PATH' WEIGHT 10) OR
 (URL_TYPE CONTAINS 'FILE' WEIGHT 0) OR
 (URL_DEPTH CONTAINS 'URLDEPTHA' WEIGHT 5) OR
 (URL_DEPTH CONTAINS 'URLDEPTHAB' WEIGHT 5) OR
 (URL_DEPTH CONTAINS 'URLDEPTHABC' WEIGHT 5) OR
 (URL_DEPTH CONTAINS 'URLDEPTHABCD' WEIGHT 5) )
```

Although it might seem this query is giving the same weight to the ROOT, SUBROOT and PATH types of URLs (all WEIGHT 10), because a ROOT term is much less frequent in the URL_TYPE column, it in effect gets a higher weight in relevance ranking because of its higher inverse document frequency (and SUBROOT has more impact than PATH for the same reason). The URL type of FILE was given WEIGHT 0, which means it did not affect the relevance calculation, but it was included so that the AND clause would match all rows.

Similarly, giving URL depths 1-4 some extra weight was found to be modestly helpful on last year's entry page finding task. Again, URLs of depth 1 (for which URLDEPTHA was included in the URL_DEPTH column) internally had a higher weight from the inverse document frequency.

For the submitted hum02uhp run, an even higher weight was given to URL_TYPE (the 3 terms of WEIGHT 10 were given WEIGHT 25). On last year's entry page finding task, the stronger URL_TYPE weights gave similar MRR scores to the lower ones.

For the submitted hum02ud run, the SearchSQL query was the same as for hum02upd except that the ALL_PROPS searches were omitted (i.e. properties and phrases in properties were not given extra weight). Note that the FT_TEXT column indexed all of the properties except for the URL of the document header.

The difference between the hum02upd and hum02up runs was in the importance of document length normalization (in general, runs ending with 'd' used "SET RELEVANCE_DLEN_IMP 500" and the others used "SET RELEVANCE_DLEN_IMP 250").

For the named page queries, no query terms were discarded (e.g. there was no expectation that discarding the words "find", "relevant" and "document" would be beneficial, unlike for some previous year's tasks). Of course, the index omitted a few stop words (e.g. "the", "by") as previously mentioned.

For the named page queries, besides linguistic expansion from stemming in the IS_ABOUT predicate, we did not do any query expansion. For example, we did not use approximate text searching for spell-correction (the organizers tried to ensure the topics were spelled correctly), and we did not use row expansion or any other kind of blind feedback technique.

SearchServer's relevance value calculation is the same as described last year [13]. Briefly, SearchServer dampens the term frequency and adjusts for document length in a manner similar to Okapi [8] and dampens the inverse document frequency using an approximation of the logarithm. SearchServer's relevance values are always an integer in the range 0 to 1000.

When multiple predicates are combined, as was done for the named page queries this year, SearchServer currently does not normalize by query length. For example, the URL_TYPE clauses of the earlier examples would have a lot less relative impact if the named page query contained 5 words instead of 1.

SearchServer's RELEVANCE_METHOD setting can be used to optionally square the importance of the inverse document frequency (by choosing a RELEVANCE_METHOD of 'V2:4' instead of 'V2:3'). The importance of document length to the ranking is controlled by SearchServer's RELEVANCE_DLEN_IMP setting (scale of 0 to 1000).

Table 1: Scores of Submitted Named Page Finding Runs

Run	MRR	%Top10	%Fail
hum02pd	0.626	82.0%	9.3%
hum02upd	0.538	75.3%	13.3%
hum02up	0.527	74.0%	11.3%
hum02ud	0.456	68.0%	16.7%
hum02uhp	0.337	51.3%	33.3%

Table 2: Impact of Submitted Named Page Finding Techniques on Reciprocal Rank

Experiment	AvgDiff	95% Confidence	vs.	2 Largest Diffs (Topic)
p (upd - ud)	0.082	(0.041, 0.124)	54-17-79	0.900 (2), 0.889 (51)
d (upd - up)	0.011	(−0.020, 0.043)	33-28-89	0.833 (36), 0.800 (41)
u (upd - pd)	−0.088	(−0.130,−0.047)	15-50-85	−1.000 (64), −0.917 (138)
h (uhp - up)	−0.190	(−0.236,−0.146)	0-87-63	−1.000 (2), −1.000 (117)

2.3 Official Results

The evaluation measures are likely explained in an appendix of this volume. Briefly, "Reciprocal Rank" for a topic is one divided by the rank in which the named page was found (using the smallest rank if there were duplicates of the named page), or zero if the named page was not found. "Mean Reciprocal Rank" (MRR) is the average of the reciprocal ranks over all the topics. "%Top10" is the percentage of topics for which the named page was found in the first 10 rows. "%Fail" is the percentage of topics for which the named page was not found in the first 50 rows.

Table 1 shows the scores of the submitted named page finding runs.

Most of the remaining tables will focus on one particular precision measure (usually reciprocal rank or average precision), comparing the scores when a particular feature (such as stemming) is enabled to when it is disabled. The columns of these tables are as follows:

- "Experiment" is the feature tested.
- "AvgDiff" is the average difference in the score.
- "95% Confidence" is an approximate 95% confidence interval for the average difference calculated using Efron's bootstrap percentile method[2] [3] (using 100,000 iterations). If zero is not in the interval, the result is "statistically significant" (at the 5% level), i.e. the feature is unlikely to be of neutral impact, though if the average difference is small (e.g. <0.020) it may still be too minor to be considered "significant" in the magnitude sense.
- "vs." is the number of topics on which the score was higher, lower and tied (respectively) with the feature enabled. These numbers should always add to the number of topics for the task.
- "2 Largest Diffs (Topic)" lists the two largest differences in the precision score (based on the absolute value), with each followed by the corresponding topic number in brackets (the named page topic numbers range from 1 to 150).

Table 2 shows the impact when isolating each technique distinguishing the submitted named page finding runs:

[2] See [12] for some comparisons of confidence intervals from the bootstrap percentile, Wilcoxon signed rank and standard error methods for both average precision and Precision@10.

- The 'p' factor (extra weight for HTML properties and phrases in properties) increased MRR 8 points. Some diagnostics of this result, including whether it holds up when URL techniques are not also used, are in the next section.
- The 'd' factor (document length importance of 500 instead of 250) made little difference.
- The 'u' factor (extra weight for URL type and depth) lowered MRR by 9 points, contrary to its substantial beneficial impact on last year's entry page task.
- The 'h' factor (even more extra weight for URL type) lowered MRR by another 19 points, even though it had a neutral impact on last year's entry page task.

2.4 Diagnostic Results

For the diagnostics, we defined a "base run" which set the document length importance to 500 and executed an IS_ABOUT search of just the content (i.e. the FT_TEXT column). For example:

```
SELECT RELEVANCE('V2:3') AS REL, DOCNO
FROM GOV
WHERE (FT_TEXT IS_ABOUT 'visiting pandas national zoo')
ORDER BY REL DESC;
```

The base run scored a 0.564 mean reciprocal rank, finding the named page in the top 10 for 75.3% of the queries while failing to find it in the top 50 for 11.3% of the queries.

Table 3 shows a comparison of various runs to the base run (always subtracting the base run's scores in reciprocal rank from the listed run). The first row compares submitted run hum02pd to the base run, like the above 'p' factor experiment but without the ill-fated URL techniques. While the average gain from properties and phrases is a little smaller (6 points), the (approximate) 95% confidence interval (1 to 11 points) indicates it is still statistically significant. Topic 64 (work/life center map) benefitted most.

The runs in the next group of comparisons in Table 3 (listed with a leading + sign) added the listed column to the WHERE clause with one-tenth the weight of the content search. For example, for the "+ TITLE" row, the query's WHERE clause was of this form:

```
WHERE
  (TITLE   IS_ABOUT 'visiting pandas national zoo' WEIGHT 1 ) OR
  (FT_TEXT IS_ABOUT 'visiting pandas national zoo' WEIGHT 10)
```

Apparently it is the extra weight on particular columns and not the use of phrases which explains most of the gain of the 'p' factor. For example, the "+ ALL_PROPS" run differs from hum02pd in that phrases are not used, but it produces similar gains. Of the HTML properties, just giving extra weight to the TITLE produces most of the gains. The URL and FIRST_HEADING also appear to be helpful for named page finding on average, while the META properties were harmful on average, but most of these latter results were not statistically significant. Note that the FT_TEXT column included the content of the other columns except for the URL column.

The next group of rows in Table 3 shows the importance of the content to named page finding. The compared runs just searched the listed column. For example, for the "TITLE" row, the query's WHERE clause was of this form:

```
WHERE (TITLE IS_ABOUT 'visiting pandas national zoo')
```

The content column (FT_TEXT) scored significantly higher, on average, than any other column by itself. The end of the confidence interval closest to zero represents a difference of at least 13 points in every case. Still, the "vs." column shows that for approximately one-sixth of the queries, just searching the columns containing the TITLE outscored content searching.

The last group of rows contains some miscellaneous experiments. The results with positive impacts were not statistically significant, but the negative impacts were. Just searching for the named page query as a

Table 3: Comparison with Plain Content Diagnostic Run in Reciprocal Rank

Experiment	AvgDiff	95% Confidence	vs.	2 Largest Diffs (Topic)
hum02pd	0.061	(0.017, 0.107)	49-16-85	0.968 (64), 0.909 (51)
+ ALL_BUT_KEYWORDS	0.058	(0.014, 0.103)	46-15-89	0.968 (64), 0.909 (51)
+ ALL_PROPS	0.055	(0.013, 0.097)	48-15-87	0.968 (64), 0.909 (51)
+ NONEMPTY_TITLE	0.047	(0.008, 0.086)	43-15-92	0.909 (51), 0.900 (38)
+ TITLE	0.042	(0.004, 0.080)	41-16-93	0.909 (51), 0.900 (38)
+ URL	0.034	(0.005, 0.066)	26-20-104	0.909 (51), 0.900 (106)
+ FIRST_HEADING	0.034	(−0.001, 0.071)	25-26-99	0.909 (51), 0.889 (138)
+ META_TITLE	−0.006	(−0.022, 0.012)	10-14-126	0.857 (143), −0.500 (101)
+ META_SUBJECT	−0.012	(−0.025,−0.001)	7-15-128	−0.500 (101), −0.500 (57)
+ META_DESCRIPTION	−0.022	(−0.052, 0.009)	19-31-100	0.889 (2), −0.857 (61)
+ META_KEYWORDS	−0.024	(−0.052, 0.002)	20-30-100	−0.750 (61), −0.667 (93)
FT_TEXT	0.000	(−0.001, 0.001)	0-0-150	0.000 (76), 0.000 (2)
ALL_PROPS	−0.212	(−0.292,−0.131)	25-81-44	−1.000 (124), −1.000 (69)
ALL_BUT_KEYWORDS	−0.218	(−0.299,−0.138)	29-79-42	−1.000 (108), −1.000 (28)
NONEMPTY_TITLE	−0.277	(−0.357,−0.196)	23-87-40	−1.000 (132), −1.000 (117)
TITLE	−0.268	(−0.347,−0.188)	23-86-41	−1.000 (92), −1.000 (107)
FIRST_HEADING	−0.445	(−0.524,−0.363)	14-114-22	−1.000 (61), −1.000 (35)
META_DESCRIPTION	−0.487	(−0.563,−0.409)	8-119-23	−1.000 (90), −1.000 (27)
URL	−0.499	(−0.577,−0.420)	10-122-18	−1.000 (92), −1.000 (43)
META_KEYWORDS	−0.502	(−0.574,−0.430)	4-123-23	−1.000 (84), −1.000 (85)
META_TITLE	−0.541	(−0.612,−0.470)	2-128-20	−1.000 (77), −1.000 (80)
META_SUBJECT	−0.562	(−0.631,−0.491)	1-132-17	−1.000 (1), −1.000 (80)
+ phrase	0.015	(−0.012, 0.045)	18-11-121	0.950 (68), 0.909 (51)
stemming off	0.013	(−0.014, 0.041)	32-16-102	−0.957 (150), 0.900 (38)
DLEN 750	0.011	(−0.016, 0.039)	30-20-100	−0.800 (85), 0.750 (139)
idf squared	−0.025	(−0.049,−0.001)	16-32-102	−0.833 (96), −0.667 (84)
DLEN 0	−0.210	(−0.262,−0.158)	7-85-58	−1.000 (36), −0.974 (96)
phrase only	−0.424	(−0.503,−0.345)	11-108-31	−1.000 (97), −1.000 (92)

phrase in the document (WHERE FT_TEXT CONTAINS 'query'), as in the "phrase only" row, significantly hurt on average, but enhancing the base run by giving a little extra weight (one-tenth) to the query as a phrase (OR FT_TEXT contains 'query'), as in the "+ phrase" row, was modestly helpful. Disabling stemming (via SET VECTOR_GENERATOR '') was only modestly helpful as per the "stemming off" row. Increasing the importance of document length normalization (via SET RELEVANCE_DLEN_IMP 750, as opposed to the base run setting of 500) didn't make much difference (as per the "DLEN 750" row), but decreasing it to 0 (as per the "DLEN 0") row significantly hurt. Increasing the importance of inverse document frequency (by using relevance method 'V2:4' instead of 'V2:3') was modestly detrimental as per the "idf squared" row. Even for the impacts which were modest on average, individual queries could have large changes in their scores as indicated by the "2 Largest Diffs" column.

3 Arabic Retrieval

The Arabic document set was the same as last year: Arabic Newswire A Corpus [1] consisting of articles from the Agence France Presse (AFP) Arabic Newswire from 1994-2000. It contained 383,872 documents, totalling 911,555,745 bytes (869 MB) uncompressed.

Table 4: Impact of Submitted Arabic Techniques on Average Precision

Experiment	AvgDiff	95% Confidence	vs.	2 Largest Diffs (Topic)
Exp (tde - td)	0.033	(0.021, 0.046)	37-13-0	0.204 (27), 0.131 (58)
Morph+Stop (td - tdm)	0.032	(0.005, 0.061)	34-16-0	0.360 (29), 0.335 (60)
Narr (tdne - tde)	0.027	(−0.014, 0.073)	27-22-1	0.794 (34), −0.551 (27)
Desc (tde - te)	0.014	(−0.011, 0.039)	29-20-1	0.348 (58), −0.311 (49)

3.1 Indexing

SearchServer (as of version 5.0) internally uses the Unicode canonical decomposition of text and, by default, does not index combining characters (accents, diacritics, etc.). For Arabic, this means by default that composite characters 0622 (alef with madda above), 0623 (alef with hamza above) and 0625 (alef with hamza below) are treated as 0627 (alef). 0624 (waw with hamza above) becomes 0648 (waw), and 0626 (yeh with hamza above) becomes 064A (yeh) (the codes and names are from the Unicode Standard [15]). All of our submitted runs both last year and this year used this default behaviour.

For the submitted runs, two different SearchServer tables called ARAB01 and ARAB01AS were created. ARAB01 was the same as last year, i.e. no stop words and no Arabic morphological normalizations were used. ARAB01AS used the stop words [9] and the experimental morphological normalizations described in section 5.2 of last year's paper [13] (which were just used for diagnostic runs last year, not submitted runs like this year).

3.2 Searching

Compared to last year, there were twice as many Arabic topics (50). They were numbered AR26-AR75 and were distributed in a file called "final_arabic02.txt". The topics contained a "Title" (subject of the topic), "Description" (typically a one-sentence specification of the information need) and "Narrative" (more detailed guidelines for what a relevant document should or should not contain). The topics were encoded in the ISO 8859-6 character set, so "SET CHARACTER_SET 'ISO_8859_6' " was executed before the searches.

Like last year, Intuitive Searching of the content was used (i.e. FT_TEXT IS_ABOUT). The statement "SET MAX_SEARCH_ROWS 1000" was previously executed so that the working table would contain at most 1000 rows. There were no experiments with phrases nor columns other than FT_TEXT.

Submitted runs humAR02tdm and humAR02td both used the Title and Description fields in the query. Run humAR02tdm searched the ARAB01 table, and run humAR02td searched the ARAB01AS table. The other settings (e.g. RELEVANCE_METHOD 'V2:3' and RELEVANCE_DLEN_IMP 500) were the same as described in section 5.2 of the final version of last year's paper [13].

Submitted run humAR02tde used query expansion from blind feedback in the same way as described in last year's paper [13]. The base run was humAR02td and the first 5 rows were used to generate broader queries.

Submitted runs humAR02te and humAR02tdne differed from humAR02tde in that the former just used the Title field in its base run, and the latter additionally used the Narrative field in its base run.

3.3 Official Results

To review the evaluation measures for topic relevance tasks: "Precision" is the percentage of retrieved documents which are relevant. "Precision@n" is the precision after n documents have been retrieved. "Average precision" for a topic is the average of the precision after each relevant document is retrieved (using zero as the precision for relevant documents which are not retrieved). "Recall" is the percentage of relevant documents which have been retrieved. "Interpolated precision" at a particular recall level for a topic is the maximum precision achieved for the topic at that or any higher recall level. For a set of topics, the measure is the average of the measure for each topic (i.e. all topics are weighted equally).

Table 5: Precision of Arabic Diagnostic Runs

Run	AvgP	P@5	P@10	P@20	Rec0	Rec30	P@R
C-td-Y01	0.180	46.4%	42.8%	41.4%	0.697	0.232	23.7%
def-td-Y01	0.209	48.0%	48.4%	46.2%	0.724	0.285	27.1%
L-td-Y01	0.286	61.6%	54.0%	49.0%	0.803	0.379	34.7%
CL-td-Y01	0.302	64.0%	56.8%	51.2%	0.833	0.402	36.2%
CLS-td-Y01	0.337	64.8%	58.4%	52.0%	0.851	0.444	38.0%
CLSE-td-Y01	0.365	64.8%	58.4%	52.8%	0.836	0.477	40.3%
C-td-Y02	0.224	31.6%	34.8%	31.6%	0.595	0.302	26.6%
def-td-Y02	0.227	32.0%	35.8%	32.5%	0.589	0.310	27.4%
L-td-Y02	0.273	42.8%	38.8%	37.1%	0.651	0.354	30.5%
CL-td-Y02	0.278	42.8%	38.4%	36.5%	0.687	0.361	31.2%
CLS-td-Y02	0.291	43.2%	39.8%	36.4%	0.712	0.374	31.6%
CLSE-td-Y02	0.323	44.8%	43.6%	39.5%	0.707	0.426	34.5%

The scores of the submitted runs are expected to be listed in the appendix of the conference proceedings. Table 4 shows the impact when isolating each technique distinguishing the submitted runs. The query expansion technique ("Exp" experiment) increased mean average precision by 3 points and was fairly consistent (small standard error), as evidenced by the narrow confidence interval. The experimental morphological normalizations plus stop words ("Morph+Stop") also increased mean average precision by 3 points, but less consistently, as evidenced by the wider confidence interval. Including the Narrative field ("Narr") increased mean average precision by 3 points, but was very inconsistent; using the Narrative often hurt the scores. Including the Description field ("Desc") increased mean average precision by just 1 point, though again was not very consistent.

3.4 Diagnostic Results

After the official runs were submitted, we used SearchServer's plug-in parser architecture to experiment with plugging in an implementation of the light algorithmic "Light8" stemmer of Larkey et al. [6]. It contained several stemming rules we were not previously using. On topic AR30, we found the light stemmer, in combination with the default dropping of combining characters, did not stem Arabic words for "satellite" to the same form, leading us to also experiment with indexing combining characters 0653 (maddah above), 0654 (hamza above) and 0655 (hamza below) via an extra line in the stopfile ("IAC="\u0653-\u0655" "). But the light stemmer explicitly dropped these combining characters if they followed 0627 (alef).

Table 5 lists precision scores for the diagnostic runs. Listed for each run are its mean average precision (AvgP), the mean precision after 5, 10 and 20 documents retrieved (P@5, P@10 and P@20 respectively), the mean interpolated precision at 0% and 30% recall (Rec0 and Rec30 respectively), and the mean precision after R documents retrieved (P@R) where R is the number of relevant documents for the topic. The following run codes were used: "Y01" (Year 2001) specifies the run used the 25 TREC 2001 topics. "Y02" (Year 2002) specifies the run used the 50 TREC 2002 topics. "L" (Light stemming) specifies the run used the light algorithmic stemmer. "C" (Combining characters) specifies that combining characters 0653-0655 were not dropped by SearchServer. "S" (Stop words) specifies the run used a table which did not index stop words. "E" (Expansion) specifies the run used query expansion from blind feedback. "td" specifies the Title and Description fields were used. "def" specifies the default settings. In particular, the "def-td-Y02" run of Table 5 is the same as submitted run "humAR02tdm" (a baseline Title+Description run not using light stemming, combining character indexing, stop words nor expansion).

Table 6 isolates the impact of various techniques on the average precision measure. All of these comparisons use "td" topics, and most of them are statistically significant at 5% level:

- The "+L" rows isolate the impact of light stemming. The impact when indexing combining characters

Table 6: Impact of Diagnostic Arabic Techniques on Average Precision

Experiment	AvgDiff	95% Confidence	vs.	2 Largest Diffs (Topic)
+L (L-def) td-Y01	0.078	(0.027, 0.134)	17-8-0	0.424 (19), 0.408 (14)
+C (CL-L) td-Y01	0.016	(0.000, 0.038)	8-5-12	0.231 (16), 0.100 (2)
+LC (CL-def) td-Y01	0.093	(0.045, 0.147)	19-6-0	0.424 (19), 0.408 (14)
+C (C-def) td-Y01	−0.029	(−0.071, 0.000)	6-9-10	−0.423 (10), −0.219 (14)
+L (CL-C) td-Y01	0.122	(0.061, 0.192)	19-6-0	0.627 (14), 0.424 (19)
+S (CLS-CL) td-Y01	0.035	(0.017, 0.056)	22-3-0	0.204 (17), 0.116 (7)
+CLS (CLS-def) td-Y01	0.128	(0.075, 0.188)	21-4-0	0.518 (19), 0.461 (14)
+E (CLSE-CLS) td-Y01	0.029	(0.009, 0.049)	17-8-0	0.138 (15), 0.116 (2)
+CLSE (CLSE-def) td-Y01	0.157	(0.102, 0.217)	23-2-0	0.559 (19), 0.377 (14)
+L (L-def) td-Y02	0.046	(0.016, 0.079)	31-19-0	0.360 (29), 0.358 (60)
+C (CL-L) td-Y02	0.005	(−0.007, 0.023)	18-14-18	0.371 (30), −0.088 (57)
+LC (CL-def) td-Y02	0.051	(0.017, 0.088)	29-21-0	0.479 (30), 0.360 (29)
+C (C-def) td-Y02	−0.003	(−0.012, 0.004)	16-14-20	−0.179 (55), 0.053 (37)
+L (CL-C) td-Y02	0.054	(0.022, 0.089)	31-19-0	0.479 (30), 0.367 (60)
+S (CLS-CL) td-Y02	0.014	(0.007, 0.020)	43-7-0	0.068 (30), 0.056 (45)
+CLS (CLS-def) td-Y02	0.064	(0.031, 0.102)	36-14-0	0.546 (30), 0.363 (60)
+E (CLSE-CLS) td-Y02	0.032	(0.016, 0.049)	35-15-0	0.294 (27), 0.147 (58)
+CLSE (CLSE-def) td-Y02	0.096	(0.057, 0.138)	38-12-0	0.599 (30), 0.424 (27)

("CL-C", i.e. subtracting the "C" run from the "CL" run) was particularly substantial on the TREC 2001 topics, where we found an increase of 0.122 (from .180 to .302). Larkey's comparable figure was 0.182 (from 0.194 to 0.376) which is inside our approximate 95% confidence interval of (0.061, 0.192). For the TREC 2002 topics, which did not exist when the stemmer was developed, we find a smaller, though still significant, increase.

- The "+C" rows isolate the impact of indexing the combining characters. When this was the only change from the default ("C-def"), the impact tended to be detrimental, perhaps because the alef forms were no longer conflated. When applied to light stemming runs ("CL-L"), the light stemmer re-conflated the alefs, and the net impact (in effect preserving composite characters 0624 and 0626) tended to be beneficial.

- The "+LC" rows show the combined impact of light stemming and indexing combining characters. Of course, the average impacts add within rounding differences (the calculations were done to 4 decimal places though just 3 are shown). However, confidence interval endpoints do not add (e.g. 0.027 plus 0.000 does not add to 0.045).

- The "+S" rows isolate the impact of using the Arabic stop word list (subtracting the "CL" run from the "CLS" run). The increases are small but fairly consistent, much like in European stop word experiments when using the full topics [12], but for the Arabic task this result also holds when omitting the Narrative.

- The "+E" rows isolate the impact of the query expansion technique (subtracting the "CLS" run from the "CLSE" run). The results are similar to the official "Exp" experiment. As the expansion terms are chosen from the first 5 rows, the first 5 rows are usually the same after expansion, which moderates how much the result can change. We haven't done a lot of work on our expansion technique and it is likely underachieving. Expanding queries generally leads to much longer processing times which can be a high price to pay for improvements in the part of the result list that users might not even look at. In practical systems, users can control the query terms themselves rather than depend on blind feedback.

- The "+CLSE" rows show the combined impact of all 4 techniques. Even the low end of the confidence intervals represent a substantial impact.

In all 9 cases in Table 6, the confidence interval for the 50 topic experiment (Y02) was narrower than the confidence interval for the corresponding 25 topic experiment (Y01). Also reassuringly, the corresponding confidence intervals always overlapped. The interval widths ranged from 1 to 8 points for the 50 topic experiments and from 4 to 13 points for the 25 topic experiments.

References

[1] Arabic Newswire Part 1, Linguistic Data Consortium (LDC) catalog number LDC2001T55, ISBN 1-58563-190-6. http://www.ldc.upenn.edu/Catalog/LDC2001T55.html

[2] Cross-Language Evaluation Forum web site. http://www.clef-campaign.org/

[3] Bradley Efron and Robert J. Tibshirani. An Introduction to the Bootstrap. 1993. Chapman & Hall/CRC.

[4] The .GOV Test Collection. http://www.ted.cmis.csiro.au/TRECWeb/govinfo.html

[5] Andrew Hodgson. Converting the Fulcrum Search Engine to Unicode. In Sixteenth International Unicode Conference, Amsterdam, The Netherlands, March 2000.

[6] Leah S. Larkey, Lisa Ballesteros and Margaret E. Connell. Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis. In Micheline Beaulieu, Ricardo Baeza-Yates, Sung Hyon Myaeng and Kalervo Järvelin, editors, Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 275-282, 2002.

[7] NTCIR (NII-NACSIS Test Collection for IR Systems) Home Page. http://research.nii.ac.jp/~ntcadm/index-en.html

[8] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, M. Gatford. (City University.) Okapi at TREC-3. In D. K. Harman, editor, Overview of the Third Text REtrieval Conference (TREC-3). NIST Special Publication 500-226. http://trec.nist.gov/pubs/trec3/t3_proceedings.html

[9] Bonnie Glover Stalls and Yaser Al-Onaizan. (University of Southern California, Information Sciences Institute, Natural Language Group.) Arabic Stop Words List. http://www.isi.edu/~yaser/arabic/arabic-stop-words.html

[10] Text REtrieval Conference (TREC) Home Page. http://trec.nist.gov/

[11] Stephen Tomlinson and Tom Blackwell. Hummingbird's Fulcrum SearchServer at TREC-9. In E. M. Voorhees and D. K. Harman, editors, Proceedings of the Ninth Text REtrieval Conference (TREC-9). NIST Special Publication 500-249. http://trec.nist.gov/pubs/trec9/t9_proceedings.html

[12] Stephen Tomlinson. Experiments in 8 European Languages with Hummingbird SearchServer^{TM} at CLEF 2002. In Carol Peters, editor, Working Notes for the CLEF 2002 Workshop. http://clef.iei.pi.cnr.it:2002/workshop2002/WN/26.pdf

[13] Stephen Tomlinson. Hummingbird SearchServer^{TM} at TREC 2001. In E. M. Voorhees and D. K. Harman, editors, Proceedings of the Tenth Text REtrieval Conference (TREC 2001). NIST Special Publication 500-250. http://trec.nist.gov/pubs/trec10/t10_proceedings.html

[14] Thijs Westerveld, Wessel Kraaij and Djoerd Hiemstra. Retrieving Web Pages using Content, Links, URLs and Anchors. In E. M. Voorhees and D. K. Harman, editors, Proceedings of the Tenth Text REtrieval Conference (TREC 2001). NIST Special Publication 500-250. http://trec.nist.gov/pubs/trec10/t10_proceedings.html

[15] The Unicode Standard Version 3.0. The Unicode Consortium. 2000. Addison-Wesley.

Arabic Information Retrieval at IBM

Martin Franz, J. Scott McCarley
IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598
<franzm,jsmc>@watson.ibm.com

February 6, 2003

1 Introduction

IBM built two systems for crosslanguage experiments with English queries and Arabic documents. One system approached translation and retrieval as entirely separate tasks: we used a machine translation system to translate the Arabic documents into English, and then did the retrieval with a standard English IR system; the other system incorporated the parameters of a machine translation model directly into an IR scoring formula. A further experiment combined both models.

For processing Arabic text, we had access to an innovative Arabic morphological analyzer, whose details will be described elsewhere. We incorporated well-known text normalizations [1] into the Arabic text processing. Our monolingual baseline system was similar to the system we have used in previous ad hoc tracks [3], and consisted of an Okapi [4] first pass followed by LCA-style [5] query expansion, applied to the normalized Arabic stems.

Translation model parameters were estimated from the U.N. parallel corpus. The English half was morphologically analyzed (as were the English queries in our submissions); the Arabic half was morphologically analyzed and text normalizations were applied. We built separate translation models relating normalized Arabic morphs to English morphs and relating Arabic words to English morphs.

2 Convolutional Model

Following the approach described in [2], we model $p(q|d)$, the probability of generating an English query q given an Arabic document d as a smoothed *convolution* of an English to Arabic translation probability with a probability of sampling Arabic text from the document. More specifically, we write

$$p(q|d) = \prod_i \left(\alpha_1 p_0(q_i) + \alpha_2 \sum_j t_w(q_i|w_j)p(w_j|d) + \alpha_3 \sum_j t_s(q_i|s_j)p(s_j|d) \right).$$

Here the q_i are the morphological stems of the English query words, w_j are the (inflected) Arabic words, and s_j are the morphological stems of the Arabic words. We estimate the sampling probabilities $p(w_j|d)$ and $p(s_j|d)$ as the appropriate token count divided by the document length. We estimate the translation probabilities $t_w(q|w)$ of English stems given Arabic words and $t_w(q|s)$ of English stems given Arabic stems using the methods of [6], with the U.N. parallel corpus as training data. These estimates are smoothed with $p_0(q_i)$, the background probability of the English word which we estimate from the English half of the U.N. parallel corpus. For each query, the top documents were selected according to $p(q|d)$, and then the top Arabic terms were selected by tf-idf (similar to [1].) The expanded query was then rescored with the Okapi formula. Although this system uses the technology of statistical machine translation, it does not result in a translation of the corpus. In particular it only predicts the "bag of words" of which an English translation of a given document would be composed. It does not try to predict the order of the words - essential for a human-readable result.

3 Document Translation

An alternative approach is to use machine translation to translate the documents into English, and then use an English monolingual retrieval system similar to our previous TREC adhoc submissions [3, 4, 5] to retrieve the documents. The Arabic-to-English statistical machine translation system heavily draws upon Arabic morphological processing modules including word segmentation, part-of-speech tagging, and a novel technique of identifying optimal word units in the source and target languages inducing a higher quality word-to-word alignments. The morphologically processed corpus is used for IBM Model 1 [6] training and decoding. Further details will be published elsewhere.

Although both of these statistical models are trained on the same training corpus, they differ in several important aspects:

(1) the convolutional model "translates" on a document-by-document basis - words arbitrarily far apart in the document influence each other's translation; the machine translation system translates on a sentence-by-sentence basis - no information propagates across sentence boundaries.

system	method	AveP	P20
ibmy02a	convolution	0.3509	0.4170
ibmy02b	doc. trans.	0.2705	0.3760
ibmy02c	merged a,b	0.3563	0.4290
ibmy02d	monolingual	0.3030	0.3820

Table 1: Results - mean average precision and precision at rank 20 of official submissions

(2) the convolutional model is based on a directly trained $p(english|arabic)$; the translation model is a source-channel model and uses $p(arabic|english)p(english)$

(3) the convolutional model sums over all possible translations of each Arabic word; the translation model makes a hard decision about each word's translation(s).

4 Results

We submitted four experiments (three cross-lingual and one monolingual) for the evaluation. The results are shown in Fig. (1) The convolutional model (ibmy02a) had noticeably better performance than the document translation (ibmy02b) model. It is not clear whether this difference is due to summing over all possible translations or to other differences in the model. We also submitted a run that combined both methods (ibmy02c) [7], for a slight improvement in performance. In Fig. (1) we show a scatter plot of the query-by-query scores of the two methods.

5 Acknowlegments

This work is supported by DARPA under SPAWAR contract number N66001-99-2-8916. The authors would like to thank Young-Suk

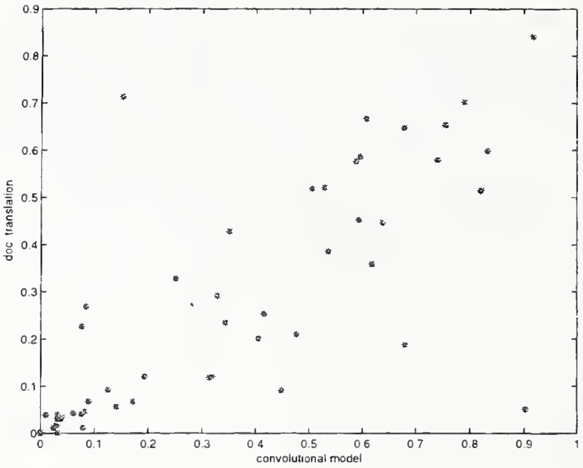


Figure 1: Scatterplot of the average precision of each query in the two IR systems.

Lee for the Arabic morphological analysis and Arabic to English translation system.

References

[1] J.Xu, A.Fraser, and R.Weischedel "TREC 2001 Cross-lingual Retrieval at BBN" in *Proceedings of the Tenth Text REtrieval Conference (TREC-10)* ed. by E.M. Voorhees and D.K.Harman, 2001.

[2] J.Xu,, R. Weischedel, and C.Nguyen, "Evaluating a Probabilistic Model for Cross-lingual Information Retrieval", In *SIGIR 2001*, pp. 105-110.

[3] M. Franz, J.S.McCarley, and R.T. Ward, "Ad hoc, Cross-language and Spoken Document Information Retrieval at IBM", in *Proceedings of the Eight Text REtrieval Conference (TREC-8)* ed. by E.M. Voorhees and D.K.Harman, p.391, 2000.

[4] S.E. Robertson, S. Walker, S. Jones, M.M. Hancock-Beaulieu, M. Gatford, "Okapi at TREC-3" in *Proceedings of the Third Text REtrieval Conference (TREC-3)* ed. by D.K. Harman. NIST Special Publication 500-225, 1995.

[5] J. Xu and W. B. Croft 1996 Query Expansion Using Local and Global Document Analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Zurich, Switzerland, pp. 4-11.

[6] P. F. Brown et al. "The mathematics of statistical machine translation: Parameter estimation", *Computational Lingustics*, 19 (2), 263-311, June 1993.

[7] J.S. McCarley, "Should we Translate the Documents or the Queries in Cross-Language Information Retrieval?", in *37th Annual Meeting of the Association for Compuational Linguistics* College Park, MD, 1999.

Topic Distillation with Knowledge Agents

Einat Amitay[1], David Carmel[1], Adam Darlow[1], Ronny Lempel[2], Aya Soffer[1]
[1]IBM Research Labs, Haifa 31905, Israel
[2]Computer Science Department, Technion, Haifa, Israel

1 Introduction

This is the second year that our group participates in TREC's Web track. Our experiments focused on the Topic distillation task. Our main goal was to experiment with the Knowledge Agent (KA) technology [1], previously developed at our Lab, for this particular task. The knowledge agent approach was designed to enhance Web search results by utilizing domain knowledge. We first describe the generic KA approach and then articulate on the use of this technology in the context of the topic distillation task. We focus mainly on the Knowledge Agent features that were used in this task. The rest of this paper is organized as follows: Section 2 describes KA in general. In Section 3 we describe how KA was used for the topic distillation experiment. Section 4 describes the obtained results. Section 5 concludes.

2 Knowledge Agents

Knowledge Agents (KA) provide domain-specific Web search in the context of dynamic domains. Knowledge agents' domains are defined by users and can thus be of any granularity and specialty. The key to the knowledge agent approach is that an agent specializes in a domain by extracting relevant information every time it performs a search and using this knowledge to improve the precision of subsequent search efforts. To this end, the KA maintains a knowledge base (KAB) that stores this information persistently. The KAB consists of a set of prominent pages in its domain, a search index of the content of these pages, and a repository of frequent terms in these pages. Each term is associated with a list of lexical affinities -- closely related terms frequently found in its proximity [2]. The agent updates the KAB continuously during search. New highly relevant pages found by the agent are inserted into the KAB; possibly replacing the place of old pages with lower utility. Pages are assigned a fitness score reflecting their relevance to the agent's domain. The KAB is used to enhance search results by automatically performing several tasks that are normally performed by users as a post-processing step after an initial unsuccessful search.

The first role the KA can perform on behalf of the user is query refinement. The KA expands the user's query by adding to each of the terms its most notable lexical affinities as found in the KAB. The advantage of this approach is that the agent's local thesaurus characterizes its domain-specific ontology and thus relations between terms are domain dependent.

The second role that the KA performs on behalf of the user is domain-specific Web search followed by shallow Web crawling. The agent applies a topological search mechanism similar to the one applied by Clever [3]. It first compiles a list of candidate result pages (root set) by sending the refined query to one or several search engines. This basic set is extended to include pages that are pointed to by pages in this set (satellite pages), pages that point to pages in the root set, and pages saved in the KAB (these are assumed to be the most authoritative information sources for the given domain). The KA ranks the retrieved pages such that the most relevant pages will be listed first. Ranking is performed based on both textual and topological aspects, utilizing information stored in its KAB. The textual score is computed by measuring the similarity of the pages to the specific query as well as to the agent's domain. The link topology score is computed using a combination of Kleinberg's mutual reinforcement algorithm [4] and stochastic link analysis [5].

The third role of the KA is local search. The KAB pages, as well as the anchor text associated with the links to the satellite pages, are indexed to form the KAB search index. The satellite pages significantly increase the number of resources accessible from the KAB. Given a query, the inverted index is used to locate the KAB pages and satellite pages that best answer this query. The results are ranked based on the similarity to the query as well as their fitness score. Since more qualitative pages in the KAB have higher fitness scores, the ranking of the results will reflect the similarity to the particular query as well as the page's authority in the KAB's domain.

The combination of a broad search of the entire Web, using general purpose search engines, with domain-specific textual and topological scoring of results, enables knowledge agents to find the most relevant documents at search time for a given query within the realm of the domain of interest. The following subsections describe the KA architecture in more detail.

2.1 The Agent's Knowledge Base

The knowledge base contains a bounded collection of ranked pages, a search index of these pages for supporting local search, and an aggregate profile of the textual content of these pages. The textual profile contains all the words that appear in the pages, after deletion of stop words and a mild stemming process, along with their number of appearances. Each word in the profile is associated with a list of its lexical affinities. The flow of pages into and out of the KAB is regulated using an evolutionary adaptation mechanism. Pages fight for the right to be included in the agent's KAB. Each page is assigned a history score reflecting its relevance to the domain through the life of the agent. A combination of the history score and the relevance score for a specific query determines which pages are inserted and removed from the KAB.

Pages can enter the KAB in two ways:

- Through explicit insertion by the user. The user may supply a set of seed pages when a new KA is created. Such seeds may come from the user's bookmarks file, or from any other collection known to the user. The user may add relevant pages to an existing KAB at any point in time. Once the KAB page limit is reached. the page with the lowest history score becomes stale and is removed from the KAB. Pages that are entered into the KAB explicitly by the user receive a high initial history score.
- Through automatic insertion by the agent. Upon completion of the search process, the t-generation history score $h_t(s)$ of each KAB page s, is updated in the following manner :

$$h_t(s) = (1-\beta_t)h_{t-1}(s) + \beta_t S(s)$$

  $S(s)$ is the score of s for the t'th search (scoring is described in the next section); $h_{t-1}(s)$ is the history score of s prior to the t'th search; β_t is a learning coefficient which controls the adaptation rate of the KAB. The learning coefficient balances the two factors which set the value of the new history score of the KAB's pages, namely the prior history score of the page, and its current specific score. The relative importance of the two components changes with the agent's age. As the number of queries performed by the agent grows, the weight of the history is increased. This reflects our confidence in KAB pages of mature agents, which have processed many queries and are therefore more likely to be highly relevant to the domain in question. Thus, we set

$$\beta_t \leftarrow \beta_0 \delta^t .$$

  β_0 is an initial coefficient value; δ is decay factor, which controls the rate of decay of the learning coefficient. The new history scores of the KAB pages are compared against the scores of new pages returned by the search. High scoring new pages may replace low scoring KAB page. Their initial history score is set to their score for the current query.

2.2 The Search Process

The search process (Figure 1) starts with the user entering a query and ends with the agent returning a ranked set of (hopefully highly relevant) pages.

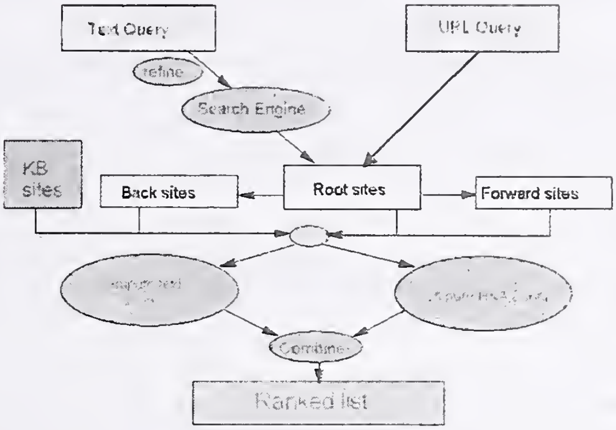


Figure 1: The search process.

2.2.1 Collecting the root set

The KA system supports two types of queries, text queries and sample-url queries. A text query is a keyword-based query such as those typically submitted to general-purpose Web search engines. The user's query can be automatically refined in the context of the agent's domain by adding to each of the query's terms its most notable lexical affinities as found in the profile of the KAB. The refined query is submitted to the user's choice of one or more search engines. The results returned by the search engine(s) to the refined query are called the root set of pages.

A sample-url query specifies a few (typically 1-5) seed urls. In sample-url queries, the user-supplied seed pages assume the role of the root set of pages, as if a search engine returned them in response to some textual query. The seed pages are read, and their combined content serves as a pseudo query for the purpose of evaluating the textual content of other pages in the search process. For the Topic distillation task, we used text queries with no query refinement.

2.2.2 Expanding the root set

The collection of pages, which at the beginning contains just the root pages, is expanded by following the hyperlinks surrounding the root pages, and by adding the pages which are stored in the KAB. The exact expansion model depends on the type of query that is being processed :

- When processing a text query, the expansion follows the scheme presented in [4] and adds two sets of pages:
    - The Backward set *B*, which contains pages that point to one or more root pages.
    - The Forward set *F*, which contains pages that are pointed to by one or more root pages.
- When processing a sample-url query, the expansion phase is more exhaustive and adds the following sets of pages to the root set:
    - The previously mentioned sets. *B* and *F*.

- The set *BF*, which contains pages which point to one or more *F*-pages.
- The set *FB*, which contains pages pointed to by one or more *B*-pages.

The breadth of the expansion is a parameter that can be set by the user. This expansion factor, which is a natural number, specifies how many pointed/pointing pages will be added to the collection for each page in each expansion stage.

We denote the entire collection of pages by *C*. Each page in *C* is assigned a textual and topological score, which are then combined as described below.

2.2.3 Computing the textual score of a Web Page

We first create a profile consisting of each word in the text query (not including stop words) along with its lexical affinities, and compute a textual similarity score for each page with respect to the query profile. For each term in the query profile, both keyword and lexical affinity, we compute a weight using a tf-idf formula. The textual profiles of the pages saved in the KAB serve as the set of documents from which the terms' document frequencies are taken.

The term weights are used to score the textual content of each page *s* with respect to a query using the following procedure. Text extracted from *s* is separated into three parts: strong, medium, regular. Strong text includes the words that appear in the title or in large font headers, medium text includes words that are either highlighted (bold, italics, etc.) or in small font headers, and regular text includes the rest. The textual score of page *s* is a weighted combination of the textual score for each text type. The textual score for each text type *t* is set to the sum of the weights of each query term appearing in the page in type *t*, normalizcd by the total number of terms of typc *t* in the page.

The agent computes an additional textual score reflecting the similarity of each page to the domain. This score is set to the dot product of the vectors of lexical affinities representing *s* and the domain. Term weights are assigned in relation to their frequency in the domain. The rationale for this is that pages that have many lexical affinities in common with the domain are most related to the domain. The two textual scores are normalized and combined to yield the overall textual similarity score of a page.

2.2.4 Computing the link topology score of a Web page

The agent builds a Web sub graph, induced by the collection of pages *C*, on which connectivity analysis is performed in order to find authoritative Web pages. The link topology score is computed by first assigning weights to the edges of the Web sub graph. Every link receives a positive weight that is set according to the anchor text associated with the link, and the "type" associated with the pages on both sides of the link (the source page and the target page of the directed hyperlink):

- Anchor Text contribution: the anchor text is the method by which the pointing page describes the destination page to surfers, and is often a good source of information regarding the contents of the destination page. Thus, anchor text that resembles the query adds weight to the link that it describes.
- Anchor Links: links that connect a KAB page with a non-KAB page (in either direction) are considered more important, since they connect a page, which is presumed to be central to the domain (the KAB page), with a page that presumably answers the specific query. Such cross-links are called anchor links, and their weight is increased by a constant.

This weighted Web sub graph is used to assign the hub and authority scores to each page. Each page receives two hub and two authority scores from which a link topology score is derived.

- Mutual Reinforcement hub & authority scores. Thesc are the hub & authority scores that the page receives when applying Kleinberg's Mutual Reinforcement algorithm [4] to the weighted graph at hand.
- Stochastic hub & authority scores. Thesc are the hub & authority scores that the page receives by applying SALSA, the Stochastic Approach for Link Structure Analysis [5] to the weighted graph at hand.

These scores are normalized and combined to form the overall link topology score.

2.2.5 Computing the overall score of a Web page

The textual score *T(s)* and link topology score *L(s)* are combined to yield the overall score of each page in *C*:

$$S(s) = \alpha_C T(s) + (1 - \alpha_C) L(s) .$$

Link topology scores are reliable only for collections in which many neighboring pages have been added around the direct search results. We therefore set the value of α_C according to the ratio between the size of the compiled collection *C* and the size of the root set. The larger that ratio, the more confidence we have in the link-based score, and the lower we set α_C . When the ratio is low, meaning that the link expansion phase did not add many pages, we increase the influence of the text-based scores by increasing α_C.

3 Topic Distillation using Knowledge Agents

For the topic distillation task we trained a knowledge agent for each topic using a query extracted from the topic title for training. We experimented with two types of training queries: 1) free text queries consisting of all title terms, and 2) an AND of all title terms (only pages that contain all title terms are retrieved). In both cases we first eliminate stop-words and stem the query terms. Note that since each topic was defined by only one query, we did not utilize the Knowledge Agent's capacity to learn a domain from a collection of queries.

While knowledge agents were originally designed to submit their training queries to several Web search engines, in this experiment agents submitted their queries to a search engine over the ".gov" collection. We used the Juru search engine [6] to index and search the ".gov" domain. The index was built similarly to the WT10G index created for the 2001 Web Track ad-hock task; each page was indexed based on its content as well as its anchor descriptions – the anchors associated with its incoming links [6].

After retrieving the root set, the set of pages retrieved by Juru for the training query, the agent retrieves the forward set, the set of pages that are pointed by the root set, and the backward set, the set of pages that link to the root set. The forward set and the backward set are retrieved using the special files attached to the ".gov" data, that provide for each page its in-links and out-links within the ".gov" domain.

Each page in the retrieved-page set is ranked by a linear combination of its textual score as returned by Juru, and its link topology score computed as described in Section 2.2.4. The combined scores are used to rank the set of pages. The top 100 pages are inserted into the agent's KAB. This set is re-ranked using a sequence of filters designed to guarantee a mixture of good sources in the top-10 list returned by the system. In the following we describe the filters used by our system. We experimented with different combinations of these filters.

We further experimented with a static (query-independent) link-based scoring mechanism that measures the quality of a page a priory at indexing time. Static scoring was used as an alternative to Hub & Authority scoring which is query dependent. Each page *p* is associated with a static score based on its number of in-links *n*.

$$St(p) = \begin{cases} 1.0 & n \geq 20 \\ \sqrt{n/20} & otherwise \end{cases}$$

At query time, the textual score of the page returned by Juru is linearly combined with the page's static score to yield its final score.

3.1 Site Compression

During our experiments with the ".gov" collection, we found that for several queries, the top-10 results returned by our system were populated mainly by pages from one or two sites. Furthermore, very short link paths usually connected the pages of each such dominant site. From a user's point of view, and according to the spirit of the WebTrack guidelines, retrieving such groups of pages is redundant – the average user will easily be able to reach all those pages by navigating from one or two good starting points in the site. When considering that groups of same-site pages usually represent the same aspect of the topic being searched, we conclude that the top-10 results should contain quality pages from a *diverse* set of sites, hopefully covering several aspects of the topic in question.

The purpose of the site compression (SC) filter is to ensure that the top-10 results of each query will indeed be diverse. Note that SC is applied in most of the popular search engines. Specifically, in our system, we did not allow the top-10 results to contain more than 3 results from any *logical site*. The notion of logical sites is used in many link analysis computations, to identify intra-domain links. Here, the logical site was derived from the name of the site as follows: the site name was stripped of the possible leading "www." and trailing ".gov". The remaining string was divided into dot-separated tokens, and the last two tokens were taken as the logical site name.

3.1.1 Technical details

Site Compression is only applied if the top-10 results contain more than 3 results from some logical site. For the purposes of the explanation below, we need the following definitions:

- The *path* to a page *p* is the URL of p, where *p*'s filename (if explicitly part of the URL) is removed.
- A link from page *p* to page *q* is called a *retreating* link if the path of *q* is a prefix of the path of *p*.
- The *neighborhood* of a page *p, N(p),* is the set of pages in *p*'s logical site that are reachable from *p* by following one or two *non-retreating* links.

The three steps of Site Compression are as follows:

1. The score of each page *p* is altered to reflect not only its own score, but also the scores of the pages of *N(p)*, in the following depth-two BFS-like process.

    a. Initially, all pages are considered to be unmarked.
    b. Let $q_1,\dots,q_k$ be the pages reachable from *p* by following non-retreating out links, ordered by non-decreasing scores. Mark pages *p* and $q_1,\dots,q_k$ and initialize the altered score of *p*, *S'(p)*, to $S(p)+0.33\sum_{j=1}^{k}2^{-(j-1)}S(q_i)$.
    c. For *j=1,...,k:* Let $w_0,\dots,w_{k(j)}$ be the yet unmarked pages reachable from q_j by following non-retreating outlinks, ordered by non-decreasing scores. Mark those pages, and increase *S'(p)* by $\sum_{t=1}^{k(j)}2^{-(t+j-2)}S(w_t)$

    The new score of each page reflects the (query specific) quality of its own content, as well as the quality of pages that are easily accessible from it by navigation within the logical site. This heuristic resembles the ideas of Marchiori in [7].
2. In this intermediate stage, the pages are first resorted according to the altered scores. Then, some pages are eliminated from contention by the following process: starting from the top ranking page, each non-eliminated page *p* eliminates from contention all the pages of *N(p)*. This ensures that the pages that remain as candidates, even if from the same logical site, are separated by browsing paths of length at least 3.
3. The pages that were not eliminated are added to the top-10 list one by one, according to the altered scores, as long as they do not violate the restriction of having no more than three pages per logical site. Violating pages are skipped over.

3.2 Title Filtering

During our experiments we additionally observed that the title of the most relevant pages contains at least one query term. The opposite was also true – the title of most irrelevant pages did not contain any query term. Based on this observation, we use the similarity of a page title to the topic title as another relevance filtering mechanism.

The title filter (TF) was designed to filter out pages from the top-10 results with a title that is not similar to the query. This filter receives as input a similarity threshold and a parameter *k* that determines how many pages to filter out. For each page in the KAB, it analyses the similarity of the page's title to the topic title (i.e., the query). The page is marked as *frail* if the similarity is lower than the given threshold. The title filter replaces the lowest *k* ranked *frail* pages in the top-10 set with the highest *k* ranked *non-frail* pages in the rest of the KAB set.

3.3 Duplicate Elimination

The ".gov" domain is rich with duplicate pages as is usually the case with Web collections. As a result, the agent returns many duplicate pages for the same topic. The duplicate elimination (DE) filter was invoked over the set of the top-10 pages to filter out duplicate results. For each page in the top-10 set, we computed its textual similarity to all other pages in the set. If the textual similarity is higher than a given threshold, the lower ranked page is filtered out from the top-10 set, replaced by the highest ranked page in the complement set. The process is terminated only after the mutual similarity of all top-10 results is lower than the given threshold.

The DE filter was invoked after the SC and the TF filters. The pages filtered out by SC and TE were assigned a very low score thus pushing them to the end of the KAB set and ensuring they will not become contenders for the top-10 list when applying subsequent filters.

4 Experimental Results

Our experiments evaluated the utility of the Knowledge Agent technology, followed by a sequence of filters for the topic distillation task, applied over the given set of 50 topics in the ".gov" domain. For each topic we trained an agent based on the topic's title and then invoked a combination of the SC, TF, and the DE filters. We submitted five runs:

1. BASE – KA search followed by SC and TF filters. The TF filter was invoked with parameter *k = 3* (meaning replacing up to 3 pages with non-relevant titles in the top-10 results).
2. T10 – the same as BASE except the TF filter was invoked with parameter *k=10.*
3. T10D – the same as T10 followed by the DE filter.
4. AP – the same as BASE except that all topic title terms were marked as "must appear" terms in the training query. As a result, the root set returned by Juru included only pages that contain all the terms of the topic title.
5. PR – in this experiment we replaced the Hub & Authority ranking applied by the agent with a static in-link based ranking. All three filters were invoked on the top-10 results.

Figure 2 shows results of our five runs for the 50 topics. The graph presents for each topic, the difference between P@10 of our runs and the median P@10 of all participants.

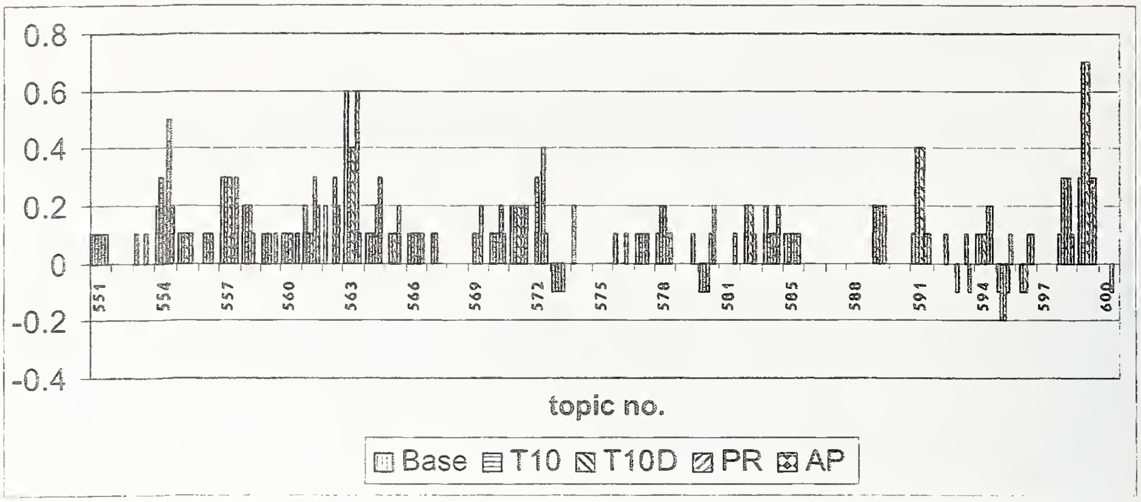


Figure 2: The difference between P@10 of our runs and the median P@10 of all participants.

As we can see, our runs achieved better results than the median for almost all topics. The following table shows the average P@10 over the 50 topics for the different runs, and the average best and median P@10 results of all participants:

BASE	T10	T10D	PR	AP	Best	Median
0.190	0.212	0.204	0.229	0.194	0.455	0.11

Table 1: Average P@10 of our runs and average-best and median P@10 of all participants.

The PR run that uses static ranking instead of link analysis achieved the best average result. This was quite surprising and contradicted our own expectations and evaluations. This is most likely due to the difference between our assessment and the official assessment, which seems to have been very strict and marked only highly *overall* authoritative pages as relevant. The T10 run achieved a better result than the BASE run, demonstrating the expected benefit of title filtering. Surprisingly however, it also outperformed T10D, which invoked the duplicate elimination filter (DE). In fact it turned out that the DE filter impaired the results. The reason is that the *trec-eval* program does not penalize duplicate pages in the top-10 set. As an example consider topic 572 –"selecting a nursing home". The T10 run returned the following three duplicate pages in its top 10: "G06-27-2524213", "G01-62-3424657", and "G01-76-0926870". All three were considered relevant by the assessor and marked as such. The DE filter filtered out two of them and left only one representative in the top-10. As a result, the number of relevant results in the top-10 was smaller after duplicate elimination and the P@10 score for this particular query was lower. This negative effect of the DE filter contradicts the guidelines for the topic distillation task that specifically stated that several pages should be returned from the same site only if they are significantly different in terms of the information they provide.

After receiving the qrel files for the 50 topics we conducted some experiments in order to better test the specific contribution of the various filters our results. Specifically, we re-created the agents, using different combinations of filters. For the Basic run. no filter was invoked. For the +SC run, the SC filter was invoked on the results of the Basic run. For The +TF(k) runs, the TF filter was invoked on the results of the +SC run, varying parameter k - the number of titles to filter. Finally, for the +DE run, the DE filter was invoked on the results of the TF(10) run. We performed the experiments using both the original KA algorithm (H&A agents) and the static ranking algorithm (PR agents). Figure 3 presents the average P@10 for each run. We can clearly see the contribution of the SC and TF filters. From these results, it is clear that the PR agents with maximal TF filtering (k= 10) indeed achieved the best results. These results demonstrate once more the negative effect of the DE filter. In fact the PR run with SC and TF(10) filtering achieves an average P@10 of 2.4, which surpasses all of our submitted runs.

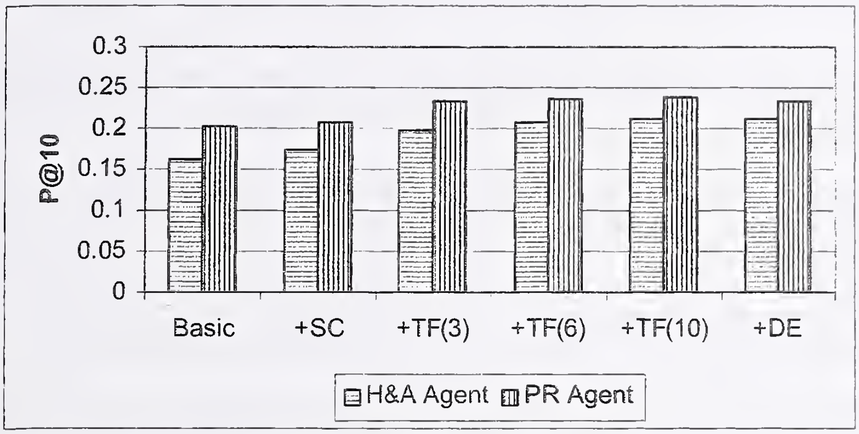


Figure 3: Filter contribution to P@10.

5. Summary

The WebTrack guidelines page [8] describes Topic Distillation as follows:

> *"Topic distillation involves finding a list of key resources for a particular topic. A key resource is a page which, if someone built me a (short) list of key URLs in a topic area, I would like to see included. ... The question is, what evidence + algorithms can be used to find such key resources? For Web searches where queries are short and users only look at the top ten results, effective topic distillation is potentially very useful".*

This report describes how Knowledge Agents can be used for a topic distillation task. We have shown how to train an agent for a specific topic. The combination of a broad search of the entire ".gov" domain with textual and topological scoring of results, enables the knowledge agent to find relevant documents for a given topic. The experiments we conducted showed that while the agent's KAB pages can be used as a basis for the result set, extra filters are required to further distill this set and surface its most valuable results.

While the SC and the TF filters improved precision significantly, the DE filter deteriorated the overall precision. This contradicts the WebTrack guidelines, which clearly state:

> *"Penalizing duplication: If your top ten consists of ten pages from the same site, judges might only reward the home page."*

We would like to suggest that for next year's task, *trec-eval* be modified in order to account for this irregularity. For example, duplicates could be grouped and only counted as one relevant result per top-10 list.

We did not make use of many of the more advanced features of the Knowledge Agent system for the topic distillation task this year. For future work, we would like to experiment with the system's domain specific query expansion and learning capabilities. The results could perhaps be further improved if, for example, agents were trained using more that one training query or by expanding the query.

References

[1] Y. Aridor, D. Carmel, R. Lempel, Y. Maarek and A. Soffer. *Knowledge Agents on the Web*. In Proceedings of the 4th International Workshop, CIA 2000, Boston, MA, July 2000. LNAI 1860, pages 15--26, Springer.

[2] Y. Maarek and F. Smadja. *Full text indexing based on lexical relations: An application: Software libraries*. In Proceedings of the 12th International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 198--206, 1989.

[3] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan. *Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text*. In Proceedings of the 7th World-Wide Web conference, 1998.

[4] J. M. Kleinberg. *Authoritative Sources in a Hyperlinked Environment*. In Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms, volume 25-27, pages 668--677, January 1998.

[5] R. Lempel and S. Moran. *The stochastic approach for link-structure analysis (SALSA) and the TKC effect*. WWW9 /Computer Networks, June, 2000, Vol. 33, No. 1-6, pages 387--401.

[6] D. Carmel, E. Amitay, M. Herscovici, Y. Maarek, Y. Petruschka, and A. Soffer. *Juru at TREC 10 - Experiments with Index Pruning*. In Proceedings of the Tenth Text REtrieval Conference (TREC 2001), National Institute of Standards and Technology (NIST).

[7] M. Marchiori, *The Quest for Correct Information on the Web: Hyper Search Engines*. WWW6/Computer Networks and ISDN Systems, 29(1997) 1225-1235.

[8] TREC-2002 Web Track Guidelines. *http://www.ted.cmis.csiro.au/TRECWeb/guidelines_2002.html*

IBM's Statistical Question Answering System – TREC-11

Abraham Ittycheriah, Salim Roukos

IBM T. J. Watson Research Center
P.O.Box 218
Yorktown Heights, NY 10598
{abei,roukos}@us.ibm.com

Abstract

In this paper, we document our efforts to extend our statistical question answering system for TREC-11. We incorporated a web search feature, and novel extensions of statistical machine translation as well as extracting lexical patterns for exact answers from a supervised corpus. Without modification to our base set of thirty-one categories, we were able to achieve a confidence weighted score of 0.455 and an accuracy of 29%. We improved our model on selecting exact answers by insisting on exact answers in the training corpus and this resulted in a 7% gain on TREC-11 but a much larger gain of 46% on TREC-10.

1 Introduction

TREC evaluations in Question Answering provide a useful application benchmark, which allows validation of a number of component technologies for which evaluation criteria are absent by providing a score for the integration of these components. Our approach since TREC-9 has been to investigate a mathematical framework under which a useful solution for question answering could be produced. We will present our model and its novel extensions below. For training our system, we collected a 4K question-answer corpus based on trivia questions and developed answer patterns for the TREC collection of documents. This corpus was used to drive a number of components we will describe below. This corpus also allowed us to investigate weights on features such as presence of the answer chunk in web documents and lexical patterns found in answers. We also describe our efforts after the evaluation to overcome the inexact answer problem and present results obtained since the evaluation.

In TREC-8 (Voorhees and Tice, 1999), the NLP community began the task of evaluating Question Answering systems and has in subsequent evaluations provided significant challenges to such systems. In TREC-9, the challenge was 50-byte answers and in TREC-10 it was definitional questions and handling rejection. To address these challenges, systems have largely adopted the architecture of predicting the answer tag of the desired answer, using a document retrieval method to select relevant documents and performing answer selection to obtain the target answer. In TREC-8 (Srihari and Li, 1999) obtained significant gains using an expanded class of entities (66). In TREC-9, improved performance was demonstrated by using boolean retrieval and feedback loops (Harabagiu and et. al., 2000). In TREC-10, use of a large number of patterns was shown to perform well for retrieving answers (Soubbotin, 2001). In TREC-11, the track agreed to several significant changes

- Exact Answers
- Single Answers
- Confidence-based Ranking of Answers

1.1 Exact Answers

Systems were required to return answers which had only the desired answer. Extra words were not accepted and their presence caused the answer to be judged inexact. Our approach of handling exact answers was to use our phrases spanned by our thirty named entity categories as well as constituent phrases of the syntactic parse of the answer (Penn Treebank style) which satisfied the answer pattern for the question. The decision to use the syntactic parse based phrases caused our system to output a large number of answers which were judged as inexact. We will describe some experiments where we changed the decision to accept only those phrases which exactly satisfy the answer pattern. Our named entity categories do not capture the differences between dates and years; nevertheless, we decided to evaluate our system without modifying the named entity categories. The named entity tags are broken along five major categories:

Name Expressions Person, Salutation, Organization, Location, Country, Product

Time Expressions Date, Date-Reference, Time

Number Expressions Percent, Money, Cardinal, Ordinal, Age, Measure, Duration

Earth Entities Geological Objects, Areas, Weather, Plant, Animal, Substance, Attraction

Human Entities Events, Organ, Disease, Occupation, Title-of-work, Law, People, Company-roles

1.2 Single Answers

In previous TREC evaluations, systems returned upto 5 answers per questions. In TREC-11, only a single answer was returned for each question. For evaluating single answers, the criteria used in this evaluation was the accuracy of the system.

1.3 Confidence-based Ranking of Answers

NIST changed the metric from the mean reciprocal rank (MRR) of previous TREC Q&A evaluations to the Uninterpolated Mean Average Precision, which we shall refer to as the confidence weighted score (CWS) defined as follows,

$$\text{CWS} = \frac{1}{N}\sum_{i=1}^{N}\frac{\#\text{ correct upto question } i}{i}$$

where N is the number of questions. This metric gives more credit to questions answered correctly at the beginning of the list. We made no specific attempt to optimize on this criteria and instead worked mostly on optimizing the accuracy of our system.

2 TREC 11 System

We model the distribution $p(c|a,q)$, which attempts to measure the c, 'correctness', of the answer and question. c can take on values of either 0 and 1 indicating either an incorrect or correct answer respectively. We introduce a hidden variable representing the class of the answer, e, (answer tag/named entity) as follows,

$$\begin{aligned} p(c|q,a) &= \textstyle\sum_e p(c,e|q,a) \\ &= \textstyle\sum_e p(c|e,q,a)p(e|q,a) \end{aligned} \quad (1)$$

The terms, $p(e|q,a)$ and $p(c|e,q,a)$ are the familiar answer tag problem and the answer selection problem. Instead of summing over all entities, as a first approximation we only consider the top entity predicted by the answer tag model and then find the answer that maximizes $p(c|e,q,a)$.

The distribution $p(c|e,q,a)$ is modeled utilizing the maximum entropy framework described in (Berger et al., 1996). We built on top of the model we used last year and those features are described in (Ittycheriah et al., 2001). The new features we investigated for this year are:

- Occurrence of the answer candidate on the web
- Re-ranking of answer candidate window using a statistical MT dictionary
- Lexical patterns from supervised training pairs

This year we submitted 3 runs, two of which measured the effectiveness of the first feature type. The last run was a feedback loop on the first run, where we included the answer string of questions which had sufficient confidence to further improve their confidence. The results are presented below in Table 1. We also provided the output of system 'ibmsqa02a' to another group at IBM for the run labeled IBM-PQSQA. The integration of our system's output with their question answering system, improved their base performance from 33.8% to 35.6%, an improvement of 5.3% in accuracy and in terms of CWS, from 0.534 to 0.586 (Chu-Carroll et al., 2002).

2.1 Training Data

We used TREC-8, TREC-9, and 4K questions from our KM database to train the model this year. This corpus represented an order of magnitude increase in size over the training data size we used last year. For each question we developed a set of answer patterns by judging several potential answer sentences in the TREC corpus. Using the answer patterns and sentences derived from the TREC corpus, we automatically labelled chunks as being correct or incorrect. The total number of chunks used in formulating the model was 207K. There were 30K instances of correct answers (though 10K were inexact) and 177K incorrect chunks.

2.2 Web Feature

The web feature was used by a number of groups last year (Clarke et al., 2001) (Brill et al., 2001) and we attempted to measure its impact on our system. We incorporated the feature as two indicators: (1) occurrence of the answer candidate in the top 10 documents retrieved from the web, (2) count of the number times the answer candidate occurred. This feature type and performing no rejection is the difference between the runs ibmsqa02a and ibmsqa02b. Removing rejection the correctly rejected questions, we note only an improvement of 7 questions by using the web based feature. Our systems have traditionally used an encyclopaedia for LCA based expansion and this may explain why the web feature is less effective in our system. We refer to this method of using the web as *Answer Verification* to differentiate it with other approaches which attempt to answer the question on the web and then look in the target document corpus for the same answer. The latter method can result in unsupported answers. We note that the number of unsupported answers is not significantly different between runs 'ibmsqa02a' and 'ibmsqa02b' (11 vs. 8) but when we used the answer strings of confident questions as feedback to the run 'ibmsqa02c', the number of unsupported answers went up significantly (18 unsupported answers).

2.3 Statistical Machine Translation Thesaurus

Generally, an answer to a fact-seeking question can be decomposed as

$$a = a_d + a_s \tag{2}$$

where a_d is the desired answer and a_s is the supporting evidence for the answer. Although words comprising the answer support are generally found in the question, words such as the focus of the question are sometimes deleted in the answer. Following our general approach of learning phenomena from training data, we used our question-answer corpus to train a Model 1 translation matrix (Brown et al., 1993). Questions were tokenized with casing information folded and answers were both tokenized and name entity tagged. A question answer pair is presented below before and after the prepreprocessing.

Q: How tall is Mt. Everest? *A: He started with the highest , 29,028 - foot Mt. Everest , in 1984*
Q: how tall is mt. everest ? *A: he started with the highest , 29,028 - foot mt. everest , in 1984 measure_ne*

We had 4K training pairs from the KM trivia database, 1.6K pairs from TREC8 and 10.7K pairs from TREC9. The latter were derived from correct judgements given to questions in those evaluations and which also came from

System	Description	CWS	Right	Inexact	Unsup	Wrong	Rej
ibmsqa02a	Base system	0.454	140 (28%)	37	11	312 (62.4%)	12/83
ibmsqa02b	No web or rejection	0.403	121 (24.2%)	43	8	328 (65.6%)	0
ibmsqa02c	Feedback loop	0.455	145 (29%)	44	18	293 (58.6%)	11/49

Table 1: Performance on TREC-11.

unique sentences in the corpus. This data was split into two and separate translation models were derived. Entries which occurred in both translation models were retained; a few of the more interesting entries are shown below in Table 2. Each word is shown with the 5 top translation candidates. For the word "who", the model prefers to see a named entity tag "person_ne" with a relative high probability. Even though the number of translation pairs is small (16.3K pairs), for the question answering application we are interested in only the most common words, which are potentially modified in the translated output of the question; rarer words have to appear identical to the form in the question. Using this additional thesaurus resource, we re-ranked the answer candidate windows (windows of text bounded by the question terms and the answer candidate) and quantized the rank into 5 bins (1,2, high, mid and low) for use in the maximum entropy answer selection module. We have not separately investigated the effect of this ranking, so details will presented in the future.

2.4 Answer Patterns

The approach described in (Soubbotin, 2001) uses patterns for locating answers. In a related work, (Ravichandran and Hovy, 2002) has shown how to extract patterns in an unsupervised manner from the web. In this work, we use the supervised corpus of question and answers to extract n-grams occurring in the answer. To specialize the pattern for a particular question type, the question was represented only by the question word and the first word to its right. To generalize the answer candidate window, it was modified to replace all non-stop question words with "<queryTerm>" and the answer candidate with "<answer>". So for the example above,

QF: how tall
MW: he started with the highest ,
<answer> <queryTerm> measure_ne

where *QF* stands for the question focus and *MW* stands for the mapped answer candidate window. Ideally, the question would be represented by more than just the word adjacent to the question word but in most cases this suffices. To overcome some of the limitations of this choice, we also chose features relating the predicted answer tag and an answer pattern. An answer pattern consists of 5-grams or larger chosen with a count cutoff. The total number of pattern features incorporated was 8.5K out 15.3K features.

3 Answer Selection

Answer selection was performed as we have in previous years with minor modifications. First, a fast-match technique of selecting answer sentences is used and top 100 sentences are selected. This phase yields sentences which have the answer- pattern in TREC-10 for 80% of the sentences. Considering the approximately 10% of questions which were to be rejected in TREC-10, the error of the sentence selector is about 10% with a list size of 100 sentences.

In order to select exact answers, we extracted all parse nodes which were noun phrases and together with all phrases which were named entities formed a candidate pool. As mentioned before, our system suffered a great deal of inexact answers in the judgement and these were mostly due to the decision to accept any phrase thus selected which had an answer pattern. Below we discuss some experiments in which a phrase is considered correct only if it contains only the answer pattern.

who	
a	$t(a \mid q)$
person_ne	0.125
,	0.010
the	0.051
.	0.046
”	0.042

haiti	
a	$t(a \mid q)$
haiti	0.076
port-au-prince	0.048
miami	0.034
people	0.021
haitian	0.018

river	
a	$t(a \mid q)$
river	0.217
the	0.081
water	0.060
location_ne	0.039
many	0.028

nuclear	
a	$t(a \mid q)$
nuclear	0.183
atomic	0.020
at	0.013
soviet	0.010
site	0.010

tall	
a	$t(a \mid q)$
measure_ne	0.056
foot	0.041
feet	0.027
-	0.017
i	0.012

team	
a	$t(a \mid q)$
team	0.099
organization_ne	0.056
game	0.030
;	0.029
their	0.023

Table 2: Translation entries for some question words.

For the training corpus of chunks with their labeled decision of correct or incorrect, we formulated features such as whether the desired named entity was found in the chunk. The features described above were added to the base model described in (Ittycheriah et al., 2001) and weights were derived using the maximum entropy algorithm. For a typical answer candidate, 50-100 features are able to fire for each decision. The answer candidate that has the highest probability is chosen for the output.

4 Rejection

For questions which are determined to have no answer in the corpus, the system was supposed to return 'NIL' as the document id. To determine which questions to reject, we employed the distribution $p(c|q,a)$ and used a threshold on the distribution. However, the system sometimes encounters events which are not sufficiently represented in the training corpus and to allow some level of control it was useful to smooth this probability with a decreasing function of chunk rank. This smooth estimate was computed as

$$p^* = (1-\alpha)p(c|q,a) + \alpha(1 - 0.1(\text{chunk\_rank}))$$

where chunk_rank was saturated at 10. This year the alpha was set to 0.2 and the rejection threshold to 0.3. The rejection threshold was optimized on the accuracy of TREC-10 questions using the TREC corpus of documents. We plot in Figure 1, the cumulative distribution function of questions with answers in the corpus and also 1.0 minus the cumulative distribution function for questions which should be rejected. The plot is for TREC-10 questions using the TREC corpus of documents for answers. We expected to reject about 80 answers in our base system and the actual run seems to have done approximately the same. The feedback loop of ibmsqa02c seems to have reduced the number of rejections and thus the precision of rejections has improved from 0.145 to 0.224 while maintaining the recall rate.

5 Analysis & Subsequent Experiments

One method of characterizing a test set is with respect to a set of answer tags. The primary difference between TREC-10 and TREC-11 is in the composition of the answer tags and these are presented for the top set of tags in Figure 2. The drastic difference between the test sets

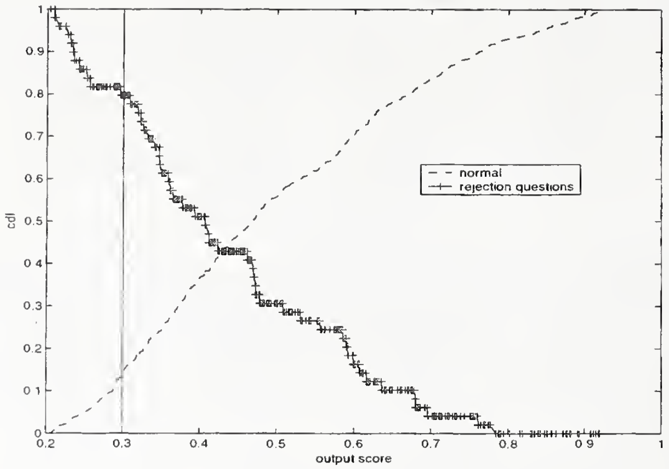


Figure 1: TREC-10 scores for normal and rejection questions.

is in the number of questions being classified PHRASE (this class represents any question which does not fall into other categories). This result reflects the reduction in definitional questions and the emphasis on exact answer questions; however, calibrating rejection rates and system strategies on TREC-10 is mismatched with the evaluation.

In order to overcome the excessive number of inexact answers produced by our system, we trained the model indicating only those phrases which exactly matched the answer pattern to be correct. As noted earlier, this is only a partial solution since some answers are now considered incorrect when they seem quite reasonable. For example in the first question of TREC-8, the answer of "Hugo Young" is now considered incorrect since the answer pattern contains only "Young". This exact match reduced the number of correct training instances about 33% (reduced from 30K to 20K where the total number of training instances is 207K); inspection of these instances indicates (a) some exact answers are now labelled incorrect, (b) majority of phrases containing the answer plus extra words are now labelled incorrect. The number of answers of type (a) is relatively small (estimated about 10% of the chunks). We then calibrated the performance of our system on TREC-11 by using the answer patterns and modifying the scoring script to accept the pattern only if

```
if ($answer_str =~ /^(\s+)?$p(\s+)?$/i)
```

The results of the system using the answer patterns are generally lower and each run seems to suffer about the same amount. Table 5 shows the results of using the new model. We emphasize that these results are obtained using the perl patterns as opposed to human judgments in the evaluation. In order to remove the effect of rejection, we modified the threshold (to 0.22 from 0.3) in the new model to output about the same number of questions rejected so that the improvement in scores is not dominated by getting only rejection questions correct. The results indicate a 46% improvement in the TREC-10 test but only about 7% gain in TREC-11. Investigating this discrepancy will be subject of future work.

In Table 3, these are answers which were accepted by the evaluation system but are now training examples for the incorrect answers. Examples of system output with the exact answer fix is shown in Table 4 with the older strings as well to demonstrate the nature of the fix. The first two examples show answers which satisfy the answer patterns exactly at test time. The last two example show errors by the system, but

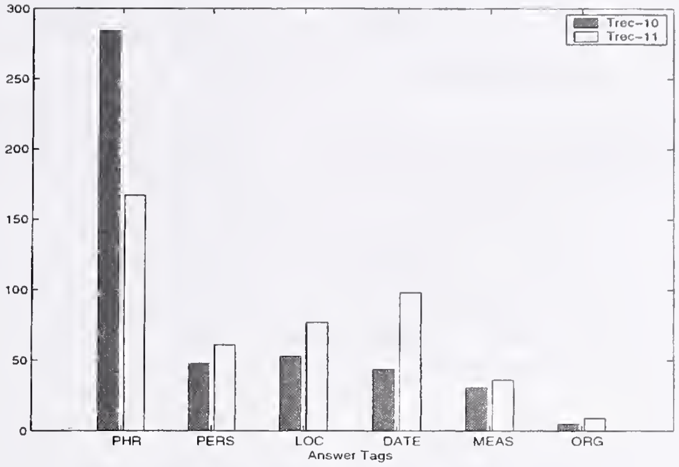


Figure 2: Comparison of answer tags between TREC-10 and 11.

What canine was made famous by Eric Knight?	Lassie Come - Home
Professor Moriarty was whose rival?	Sherlock Holmes' nemesis
What is Francis Scott Key best known for?	write the Star Spangled Banner

Table 3: Training data instances which are rejected for the exact answer fix.

Qnum	Question	Old Answer	Answer
1059	What peninsula is Spain part of?	position on the Iberian Peninsula	Iberian Peninsula
1215	When was President Kennedy shot?	shot on Nov. 22 . 1963	Nov. 22 , 1963
1316	What was the name of the plane Lindbergh flew solo across the Atlantic?	Spirit of St. Louis	Charles A.
1348	How cold should a refrigerator be?	28 degrees Farenheit	soda ice cold

Table 4: TREC-10 QA pairs before and after the exact answer fix.

System	Description	CWS	Right	Wrong	Rej
trec10-perl	Base system	0.289	92	408	10/76
trec10-perl	Exact answer fix	0.423	127	373	16/92
ibmsqa02a-perl	Base system	0.438	134	366	12/83
ibmsqa02a-perl	Exact answer fix	0.469	144	356	4/52

Table 5: Experimental results using perl-patterns since TREC-11 evaluation.

overall the system was able to produce more answers which satisfied the exact match criteria.

6 Conclusions and Future Work

In TREC-11, our method of selecting which candidates were exact answers did not satisfy the exact match criteria of the evaluation. We have since modified our system to extract exact answers and retrained the system. We incorporated two novel concepts (a statistical machine translation thesaurus and lexical patterns derived from supervised question-answer pairs) since last year.

In TREC-11, although we thresholded the distribution $p(c|q, a)$ to reject answers, this we recognize as being deficient in the following sense. We should recognize a question as not having an answer in the corpus by taking into consideration all the answers found and not just the top ranking answer.

7 Acknowledgement

This work is supported in part by DARPA under SPAWAR contract number N66001-99-2-8916.

References

Adam L. Berger, Vincent Della Pietra, and Stephen Della Pietra. 1996. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. 2001. Data-intensive question answering. *TREC-10 Proceedings*, pages 393–400.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

Jennifer Chu-Carroll. John Prager, Christopher Welty, Krzysztof Czuba, and David Ferruci. 2002. A multi-strategy and multi-source approach to question answering. *To appear in TREC-11 Proceedings.*

C.L.A. Clarke, G.V. Cormack, T.R. Lynam, C.M. Li, and G.L. McLearn. 2001. Web reinforced question answering (multitext experiments for trec 2001). *TREC-10 Proceedings*, pages 673–679.

Sanda Harabagiu and et. al. 2000. Falcon: Boosting knowledge for answer engines. *TREC-9 Proceedings*, pages 50–59.

Abraham Ittycheriah, Martin Franz, and Salim Roukos. 2001. IBM's statistical question answering system – trec-10. *TREC-10 Proceedings*, pages 258–264.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the ACL*, pages 41–47.

M. M. Soubbotin. 2001. Patterns of potential answer expressions as clues to the right answers. *TREC-10 Proceedings*, pages 293–302.

Rohini Srihari and Wei Li. 1999. Question answering supported by information extraction. *TREC-8 Proceedings*, pages 75–85.

Ellen M. Voorhees and Dawn M. Tice. 1999. The TREC-8 question answering track evaluation. *TREC-8 Proceedings*, pages 41–63, Nov.

A Multi-Strategy and Multi-Source Approach to Question Answering

Jennifer Chu-Carroll **John Prager** **Christopher Welty**
Krzysztof Czuba **David Ferrucci**
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598, USA
jencc,jprager,welty,kczuba,ferrucci@us.ibm.com

1 Introduction

Traditional question answering systems typically employ a single pipeline architecture, consisting roughly of three components: question analysis, search, and answer selection (see e.g., (Clarke et al., 2001a; Hovy et al., 2000; Moldovan et al., 2000; Prager et al., 2000)). The knowledge sources utilized by these systems to date primarily focus on the corpus from which answers are to be retrieved, WordNet, and the Web (see e.g., (Clarke et al., 2001b; Pasca and Harabagiu, 2001; Prager et al., 2001)). More recent research has shown that introducing feedback loops into the traditional pipeline architecture results in a performance gain (Harabagiu et al., 2001).

We are interested in improving the performance of QA systems by breaking away from the strict pipeline architecture. In addition, we require an architecture that allows for *hybridization* at low development cost and facilitates experimentation with different instantiations of system components. Our resulting architecture is one that is modular and easily extensible, and allows for multiple *answering agents* to address the same question in parallel and for their results to be combined.

Our new question answering system, PIQUANT, adopts this flexible architecture. The answering agents currently implemented in PIQUANT vary both in terms of the strategies used and the knowledge sources consulted. For example, an answering agent may employ statistical methods for extracting answers to questions from a large corpus, while another answering agent may transform select natural language questions into logical forms and query structured knowledge sources for answers.

In this paper, we first describe the architecture on which PIQUANT is based. We then describe the answering agents currently implemented within the PIQUANT system, and how they were configured for our TREC2002 runs. Finally, we show that significant performance improvement was achieved by our multi-agent architecture by comparing our TREC2002 results against individual answering agent performance.

2 A Modular and Extensible QA Architecture

The architecture adopted by our PIQUANT system, shown in Figure 1, defines several basic roles that components of a QA system can play. The definition of each role includes a consistent interface that allows components implementing that role to be easily plugged into the system. This architectural approach is not simply to facilitate good software engineering in a group, but it allows hybridization at a fairly low development cost, and it also facilitates experimentation based on the choices available within the different component roles.

The main components of our architecture are briefly described as follows:

1. **Question Analysis** components analyze questions to produce information consumed by other components in the form of a QFrame. Information contained in the QFrame should minimally include a question type that would help guide the selection of one or more answering agents (see below) appropriate for addressing the question. A QA system typically has one question analysis component, but may possibly have as many as one per answering agent.

2. **Answering Agent** components implement answer finding strategies given the results of question analysis and a knowledge source. These may be as simple as composing a bag-of-words query for document/passage retrieval, or as complex as breaking the question into sub-questions and consulting multiple knowledge sources. We expect QA systems to have multiple answering agents that pursue different strategies in parallel, which we believe to be an important feature of our architecture: not only can we experiment with different question answering strategies and knowledge sources, but with combining them as well.

3. **Answer Resolution** components combine the results of multiple answering agents into a single rank-

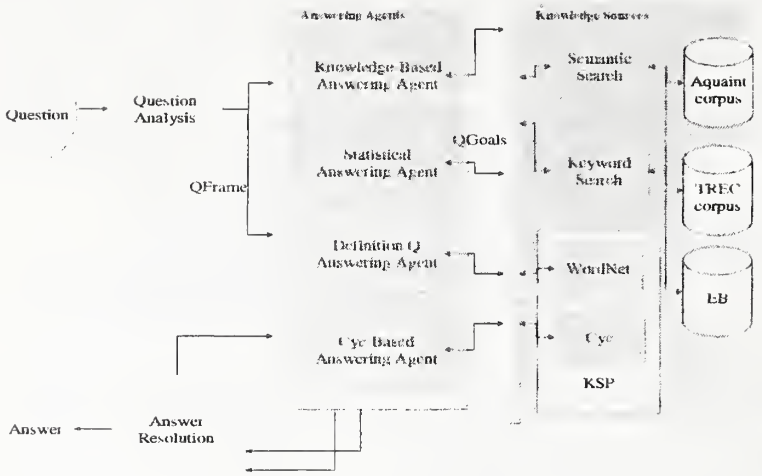


Figure 1: PIQUANT's Architecture

ing. These components may simply perform ranking over the combined set of answers of all answering agents, or may do something more complex such as feeding answers from one agent back into others. Ultimately, the final answers of a QA system are provided by this component, so there is only one such component in any QA system.

4. **Knowledge Source Adapter** components insulate the other components of the QA system that consult knowledge sources from the multitude of data formats, access mechanisms, representation languages, reasoning services, and ontologies that consumers of existing structured knowledge sources must be acutely aware of.

An obvious benefit of our component-based approach is that we can easily experiment with and compare different techniques for filling these roles by keeping the rest of the components of the QA system fixed and changing only the components that implement the techniques we wish to compare. Thus we could, for example, measure the overall impact on QA performance of using statistical vs. rule-based annotators, or using machine learning vs. rule-based answering agents. In addition, as noted above, we can often *combine* the strengths of different techniques to improve overall performance, which will be the focus of this paper.

3 A Multi-Agent Approach to Question Answering

Answering agents that can be adopted for QA may differ along various dimensions. One such dimension is the type of knowledge source from which answers are extracted, which may include unstructured text resources or structured knowledge sources such as Cyc (Lenat, 1995) or WordNet (Miller, 1995). Even when two answering agents consult the same knowledge source, they may adopt different processing strategies. For example, existing question answering systems vary greatly, from utilizing primarily knowledge-driven components, e.g., (Harabagiu et al., 2001; Prager et al., 2000) to adopting mainly statistical methods, e.g., (Ittycheriah et al., 2001; Ravichandran and Hovy, 2002).

We have so far integrated into our PIQUANT system answering agents that utilize both structured and unstructured knowledge sources. For the latter class, we have incorporated two answering agents adopting fundamentally different processing strategies. This section describes each of these answering agents, as well as how their answers are combined to formulate the system's final answers.

3.1 Agents Based on Unstructured Information

Perhaps motivated by the TREC QA track, the vast majority of existing question answering systems adopt a large text corpus as their information source. Additionally, while many such systems adopt a classic pipeline architecture, each typically employs a different approach in instantiating its components. Currently, we have incorporated two text-based answering agents into PIQUANT, one utilizing a primarily knowledge-driven approach and the other adopting statistical methods. These two answering agents have performed quite comparably in past TREC QA tracks.

3.1.1 Knowledge-Based Answering Agent

Our first answering agent utilizes a primarily knowledge-driven approach to question answering, based on Predictive Annotation (Prager et al., 2000; Prager et al., 2003). A key characteristic of this system is that potential answers, such as person names, locations, and dates, in the text corpus are *predictively annotated*. In other words, the text corpus is indexed not only with keywords, as is typical for most search engines, but also with the semantic classes of these pre-identified potential answers.

During the question analysis phase, a rule-based mechanism is used to determine one or more of about 80 semantic types of the candidate answer, along with a set of keywords. A weighted search engine query is then constructed from the keywords and the candidate semantic classes. The search engine then returns a small (typically 10-passage) set of 1-to-3-sentence passages based on the query. The candidate answers in these passages are identified and ranked based on three criteria: 1) match in semantic type between candidate answer and expected answer, 2) match in weighted grammatical relationships between question and answer passages, and 3) answer frequency.

3.1.2 Statistical Answering Agent

The second answering agent used in PIQUANT is the statistical question answering system of Ittycheriah et al. (Ittycheriah et al., 2001). This statistical answering agent is also based on the pipeline architecture; however, instead of adopting rule-based mechanisms, it utilizes a maximum entropy approach for training system components.

In question analysis, one of a set of 32 potential answer types is selected based on features such as words, POS tags, bigrams, and question word markers. The search module adopts a two-pass approach in which high scoring passages from an encyclopedia are used to augment the query terms, which are then used for search against the TREC corpus. The search engine returns a large set of passages (100) for further consideration. Named entities and their semantic types are identified from these passages, again using a maximum entropy based mechanism, and a confidence value computed for each named entity based on its likelihood of being a correct answer to the given question.

3.2 Agents Based on Structured Knowledge Sources

It has been previously established that finding the answers to questions in structured knowledge sources such as WordNet and including the answer in a bag of words can improve accuracy (Prager et al., 2001). We have expanded on this notion in two ways, first by adapting a wide variety of knowledge sources into our QA system, and second by handling the case of numerical answers in post-hoc answer filtering.

3.2.1 Knowledge Server Portal

For certain classes of routine fact-seeking questions, such as populations and capitals of geo-political entities, the answering agent recognizes a number of ways of asking these questions and formulates a query to a *structured knowledge source*. These knowledge sources include public databases such as the US Geological Survey, websites with data in formatted tables from websites such as `http://www.UselessKnowledge.com`, public domain lexicons such as WordNet, and the Cyc knowledge base.

Each of these knowledge sources is maintained by external groups and is out of our direct control. Each source has data in a different format, requires a different access mechanism, is expressed in a different representation language, provides different reasoning services, and assumes a different ontology. In addition, this external control means that any of these formats, access mechanisms, etc., may change, and of course adding new knowledge sources introduces a new set of choices to be aware of.

Rather than require that each answering agent understand all these dependencies in order to use the knowledge sources, we have isolated the role of adapting external structured knowledge sources and presenting a consistent set of choices to all the QA components through a set of knowledge-source adapters. We refer to the system component that provides access to these knowledge-source adapters as the *knowledge server portal (KSP)*. The adapters provided by KSP support the set of queries the question analysis component is capable of recognizing, such as "What is the capital of Syria?" or "What is the state bird of Alaska?", and are responsible for composing the proper query to the knowledge sources that may have the answer. The answering agent then may formulate a query that includes the answer as a search term similar to (Prager et al., 2001).

3.2.2 Cyc Sanity Checker

For certain questions, in particular questions that have numerical answers, adding the answer as a search term is not effective, because there are innumerable variations on the way the number may be expressed in the corpus. Populations, for example, vary over time by a significant amount, and are usually in the millions. For a question like, "What is the population of Maryland?", knowing that the latest figure for the population of Maryland is 5,296,486 does not quite help us search the corpus, because we are almost guaranteed that precise number will not appear. It could be expressed as "5 million", "5.1 million", "5.3 million", or "5,200,390", etc. This process is complicated further when unit conversions are required, as in the question, "How big is Australia?" In addition to having to find a number in the vicinity of "1 million square miles", we also need to account for the fact that the passage may talk about square kilometers, or acres. Instead of folding the known answer into the query in cases like this, we allow the question answering system's regular procedure to generate a set of candidate answers first, and check them to be within some experimentally determined range of the answer the knowledge source provides.

We have implemented the validation of answers with numerical values using an interface to Cyc called the Cyc sanity checker. The sanity checker is invoked with the expected semantic type of the answer (such as POPULATION in the first example above), the focus of the question ("Maryland"), and the system's proposed answer ("X people"). It returns one of the following verdicts: "in range", if the proposed answer is within a certain "fudge factor" (currently 10%) of the value in Cyc's knowledge base, "out of range", if the value falls outside of the acceptable range of values, or "don't know", indicating that Cyc either has no information about the focus itself, or about the particular attribute in question about the focus.

3.3 Answer Resolution — Putting it All Together

We have described four independent answering agents currently incorporated into our multi-agent architecture. With the exception of the Cyc sanity checker, which is invoked as a post-hoc filtering process for rejecting unreasonable answers, the other three answering agents actively contribute potential answers to a given question. It is then the task of the answer resolution component to determine how the various answers proposed by each answering agent should be combined and reconciled.

Because of the TREC requirement that all answers be justified by passages from the given corpus (henceforth referred to as the AQUAINT corpus), we feed potential answers given by KSP back into the search process to identify relevant passages in a process similar to that described in (Prager et al., 2001). These passages typically contain answers identified by KSP, as well as relevant question terms; thus, they are good candidate passages for locating justification for the answer provided by KSP in the reference corpus. Because of this answer feedback mechanism, all answering agents produce relevant passages and ranked candidate answers in a uniform fashion, simplifying the answer resolution process.

Currently, PIQUANT's answer resolution component allows for merging at two different points in the pipeline as follows:

- Passages proposed by multiple answering agents can be combined to feed through the answer selection component of our knowledge-based answering agent.
- Candidate answers proposed by different answering agents can contribute to determining PIQUANT's final output.

In addition to determining the answer to a given question, the answer resolution also computes a *confidence value* indicating the system's certainty in the given answer being a correct answer to the question. This confidence value can then be used for ranking system responses for TREC submissions.

4 Recognizing When the System Does Not Know

To make the task more realistic, the test set for the QA track contains a number of questions for which no answer can be found in the document collection, as verified by NIST (we call such questions "NIL questions" or "no-answer questions"). To simplify the task of detecting no-answer questions, we reduce it to the problem of finding the questions for which we can reasonably assume that the system was not able to find a correct answer. This is a much weaker condition since it is dependent on the answer search strategy the system implements, i.e., there might be other strategies that would be successful at finding an answer. It can, however, be implemented easily by setting a threshold on the confidence value that is assigned to a question by the answer resolution module.

We implemented two strategies for determining which questions had no answers: a knowledge-based strategy and strategy based on confidence processing. The knowledge-based strategy makes use of KSP and is evoked for questions that were classified as appropriate for KSP look-up. If KSP was able to provide an answer to such a question and the answer string could not be found in the collection, we assumed with high confidence that the question is a NIL-question. Since KSP has only recently been integrated into the system and the number of questions that are referred to it is still limited, this NIL-assignment strategy applied to only two questions in the final submission.

In our confidence-based approach, we adopted a two-stage processing strategy for detecting and ranking no-answer questions. The first stage detects which questions are likely to have no answer in the collection by comparing their scores with a trained confidence threshold. The second stage takes care of the proper ranking of questions likely to have no answers by increasing their rank.

In order to train the NIL assignment algorithm, we ran our system on the TREC-10 question set and plotted the distribution of different question types in the final ranking. We marked the questions that did not have an answer according to NIST, the questions for which the system produced a correct answer, and the questions for which the system's output was wrong. The resulting plot is in

```
                                                    NIL  CORRECT
         xxxxxxxxxxxxxx.xx.xxxxxxxxxxxxxx.x..xx.xx   0     35
xxxxxx-x.-x.xxxxxxxx..x-xxxxxxxxxx.xxxxx.x.xxx.-xx   4     38
xx.....x.-xx.....xx....x.xx.x..xxx.xx...xx.x..xx.x   1     22
.-...x.xx-..x..x.xx....xx.x...xx.....x..xxx....xx.   2     18
........x....x..xxxx...x...xx....xxxxx--......xxx.   2     17
..x.xxx...-x-...xx.....x...xx--.xx-....xx..x..x...   5     16
..x.x.-......x....x.x-.x.xx...-x-x-x-...-..x-x.x.x   8     15
x..-x.....x.x.....-..........-...-..x.-....-..x...   6      6
.x--......xx....-.-..x.-....-.-..x............--...   9      5
-.-.-..--...-x.xx....-.-x......-.....-..-...-.x.-.  13      5

x correctly answered question
. incorrectly answered question
- NIL question according to NIST
```

Figure 2: TREC-10 training data for NIL assignment

Figure 2. It represents the 491 Trec-10 questions (9 questions were thrown out, see (Voorhees, 2001)) split into blocks of 50 (based on TREC-10 we expected approximately 50, or 10%, of the questions to have no answers). Next to each block we plotted the number of questions within that block that were answered correctly and the number of NIL questions within that block. As can be seen in Figure 2, the numbers change almost monotonically, which suggests that the confidences produced by the system could be a reasonably reliable indicator of the system's performance on a given question.

According to Figure 2, the final two blocks contain more NIL questions than correctly answered questions. This means that changing the system's answer to NIL for all the questions in these two blocks will produce a net gain of 12 correctly answered questions. It will also change the incorrect answers to NIL for 68 questions, which is valuable from the user's point of view, assuming that "NIL" could be interpreted as "I don't know."[1]

Based on this analysis we manually picked (on the training data) a confidence threshold that would allow us to select the 100 lowest ranked questions. We used the same threshold on the test set and changed whatever answers the system found for the questions below the threshold to NIL.

We also looked at how the average precision changed within a 50-question window as we moved it by one question at a time down the ranking, and we found the trend to be close to monotonic. Since changing the answers in the final two blocks to NIL caused the average precision within these blocks to increase, we decided to move the two blocks higher in the ranking to the rank with the same average precision. We computed the difference in confidence value between the answer at the target rank and the highest ranking answer in the NIL-block. This difference was then added to the confidence values of all NIL answers in our runs submitted to TREC.

5 Performance Evaluation

5.1 Experimental Setup

For the 2002 TREC QA track, we submitted three runs, each evaluating a different aspect of PIQUANT's multi-strategy, multi-source architecture. These three runs were set up as follows:

1. Run "IBMPQ" exploits the multi-source aspect of PIQUANT with the knowledge-based answering agent. However, instead of only searching in the AQUAINT corpus for relevant passages, we adopt two other supporting corpora: the corpus used in the TRECs 8-10 QA tracks (henceforth referred to as the TREC corpus) and a subset of the Encyclopedia Britannica. A corpus plays a supporting role when candidate answers found in that corpus can be used to boost the confidence of the same answer found in the main corpus, but the corpus cannot propose new answers not found in the main corpus.

2. Run "IBMPQSQA" exploits the multi-strategy aspect of PIQUANT by incorporating results from the SQA statistical answering agent made available to us by lttycheriah and Roukos (Ittycheriah et al., 2001). The knowledge-based answering agent was configured to retrieve relevant passages from the AQUAINT and TREC corpora. Additionally, the top 10 passages with the correct answer type retrieved by the statistical answering agent were also considered. PIQUANT's answer resolution component then selects and ranks answers based on passages from the three answering agents/sources. Once the top answer for each question is determined, PIQUANT's confidence score for the answer is adjusted based on the answer given independently by the statistical answering agent. A large boost in confidence is given to identical answers proposed by both systems, whereas a small boost in confidence is given to partially overlapping answers.

3. Run "IBMPQSQACYC" examines the effect of the Cyc sanity checker as a post-hoc filtering process. The system is configured exactly as in run "IBMPQSQA" with the following exception. Prior to determining the top answer for each question, PIQUANT repeatedly invokes the Cyc sanity checker with a semantic representation of the question and the topmost uneliminated candidate answer as long as the sanity checker deems the given answer "out of range". PIQUANT then eventually selects its most confident answer acceptable to the sanity checker. Note that if this top ranked answer is considered "in

[1] If the systems participating in the competition were penalized for providing incorrect answers, the questions in the third-to-last block could also be changed to NIL with no net gain in the number of correctly answered questions but significantly fewer potentially confusing answers.

range" (as opposed to "don't know"), its confidence is given a strong boost, as it is independently validated by a structured knowledge source.

After PIQUANT generates the answer to each question and its associated confidence, the NIL-assignment process discussed in Section 4 is invoked. As a result, answers with low confidences were changed to NIL and their confidences slightly increased.

5.2 Results and Analysis

5.2.1 Results of Submitted Runs

Table 1 shows the results of our three runs both in terms of percent correct and average precision. For comparison purposes, it shows, in addition, the performance of the statistical answering agent submitted independently to the same track (ibmsqa02a) (Ittycheriah and Roukos, 2002), as well as the performance of the knowledge-based answering agent using only the AQUAINT corpus (PQ single).[2] A comparison between the results for PQ single and IBMPQ shows the impact of the multi-source aspect of PIQUANT. Our results show that by attempting to identify supporting evidence from two additional corpora. the system achieved 19.9% relative improvement in the percentage of correct answers, and the average precision score improved by 14.6%. A comparison of the results for runs IBMPQ. ibmsqa02a, and IBMPSQA shows the contribution of adopting multiple strategies for question answering in PIQUANT. Although the percentage of questions answered correctly improved for both systems (from 33.8% for IBMPQ and 28% for ibmsqa02a to 35.6% combined), the gain in average precision is much more substantial (9.7% relative improvement compared to IBMPQ). This confirms our intuition that when answering agents (semi-)independently arrive at the same answer, we can be more confident that the answer is a correct one. A comparison of the results for runs IBMPQSQA and IBMPQSQACYC illustrates the impact of the Cyc sanity checker. Although the impact as shown is very minimal, we should note that because of the limitations in PIQUANT's current question understanding capabilities, the sanity checker was invoked only for 3 out of the 500 questions (although there were several more questions which fit the profile but were not detected as such). Additionally, out of the 3 questions, Cyc only had knowledge about one of them, *"What is the population of Maryland?"* It is the effect of sanity checking on this question that led to the improved performance for our last run. PIQUANT's top ranked answer for this question in run IBMPQSQA was "50,000", from the sentence "Maryland's population is 50.000 and growing rapidly." This would otherwise be an excellent answer if it were not for the fact that the article from which this passage is extracted discusses (the Maryland population of) an exotic species called nutria. By employing sanity checking, however, PIQUANT was able to consider that answer "out of range", and return an initially lower-ranked correct answer "5.1 million" instead with high confidence.

5.2.2 Effects of NIL Assignment

In our best submission run (IBMPQSQACYC), the confidence-based NIL-assignment strategy resulted in 147 NIL answers, which was more than we anticipated. This is due to the generally lower confidences on a new question set. The system correctly assigned NIL to 29 out of 46 questions, which translates to a recall of 0.630 and precision of 0.196. By assigning the NIL answers, the system changed 9 correct answers incorrectly to NIL, which gave us a net gain of 20 questions (given the answer pattern set currently available to us). The questions for which the answer was changed to NIL were then moved to rank 288, which resulted in a very minimal (below 0.5%) improvement in the final average precision score.

5.2.3 Analysis of the Average Precision Metric

If the same scoring method had been used this year as in previous TREC QA tracks, the mean reciprocal rank (MRR), exercised over a single answer per question would amount to a simple count of number correct. However, in order to begin to tackle the issue of answer reliability, answers this year were returned by participants in decreasing order of system confidence (although no numerical values representing confidence were returned). The systems' final scores were evaluated by Average Precision, the average being computed over the first answer, the first two answers, the first three answers, and so on up to the whole set. Clearly, this gives considerably more relative weight to the earlier answers, and considerably less to the last answers. The contribution c_k of a correct answer in position k out of N questions in total is given by $ln(\frac{N+1}{k+1}) \leq Nc_k \leq ln(\frac{N}{k}) + \frac{1}{k}$.

The plot in Figure 3 shows this contribution, in units of 1/500, for positions 1 to 500 for a set of 500 questions. Relative to a score of approximately 1 unit for the greater part of the range, the contribution of the first position is nearly 7, indicating how important it is for systems to sort their submissions well.

Another view of the evaluation space introduced by the Average Precision metric is presented in Figure 4. The diagonal line and "cloud" represent what happens with no attempt to sort the results. The solid line in the center of the cloud is the ideally-uniformly-distributed case (i.e. if 1/3 of the answers are right, the submitted list goes ...RWWRWWRWWRWW...), and the

[2] The results for PQ single were obtained by manual evaluation by one of the authors with reference to available judgments by NIST accessors and answer patterns made available by Ken Litkowski.

	IBMPQ	IBMPQSQA	IBMPQSQACYC	ibmsqa02a	PQ single
% Correct	33.8%	35.6%	35.8%	28.0%	28.2%
Avg Prec	0.534	0.586	0.588	0.454	0.466

Table 1: PIQUANT's TREC 2002 Run Results

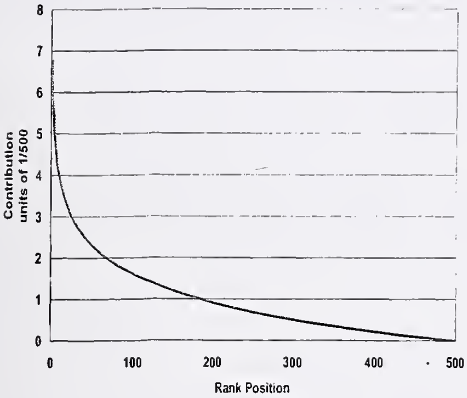


Figure 3: Contribution of Correct Answers to Average Precision Score

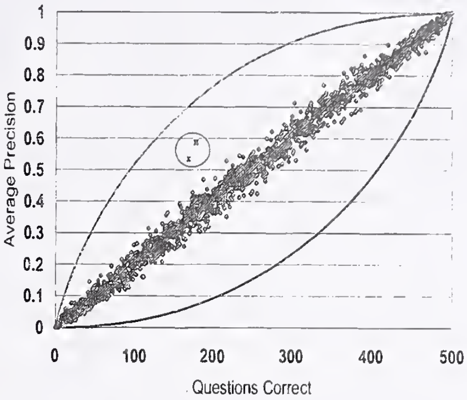


Figure 4: Upperbounds and Lowerbounds for Average Precision Scores

cloud is a simulation of randomly-distributed rights and wrongs, given the number of correct answers. The width of the cloud approximately represents a 3-standard-deviation spread. The upper curve is the optimal case (e.g. RRRRRR......WWWWW), while the lower curve is the pessimal case (i.e. all the right answers are sorted to the end.)

5.2.4 Ranking Ability

The circled points in the middle of Figure 4 represent our TREC runs. The maximum possible score, represented by the upper curve, for n correct out of N is approximately $\frac{n}{N}(1 + ln\frac{N}{n+1})$. By examining how far up a virtual vertical line from the diagonal (*expected*) to the upper curve (*max*) a plotted point (*actual*) lies, one can see how well the system sorted its answers for submission - i.e. how well it knows what it knows. This fraction, which we call the *Ranking Ability*, can be computed as $\frac{actuol-expected}{max-expected}$. In the case of our best run, we scored 179 questions correct (35.8%), for which the expected unsorted average precision is 0.358. The maximum possible average precision is 0.726 for this number correct, based on the above formula. Our score of 0.588 represents a ranking ability of .625, indicating a good correlation of confidence and correctness. The top 15 submissions are shown in Table 2, sorted by ranking ability.

6 Conclusions and Future Work

We have presented here the first quantitative results from our new PIQUANT question answering system. PIQUANT exploits a multi-strategy and multi-source approach to QA, enabling not only the best approach to be taken on a per-question basis, but the use of mutual reinforcement when multiple agents or sources are used simultaneously. Based on our submissions to TREC and their results, we have shown significant improvements achieved by our approaches over baseline systems. First, we have shown an 14.6% relative gain in average precision score with multiple corpora over a single one, and a further 9.7% relative gain by adding a statistical answering agent. Second, we have identified an effective method for assigning NIL answers to questions based on the confidence values generated by our system. This method identified 63% of all no-answer questions in the test set with minimal false negatives. Third, we have shown that a multi-agent approach to question answering allows us to achieve a good correlation of confidence values and correctness. Our average precision of 0.588 on 179 correct questions achieved 62.5% of the gain achievable by sorting, a significant improvement over the baseline of random sorting.

We have only just begun to incorporate a knowledge

Submission	AP	% Correct	Ranking Ability
limsiQalir2	.497	26.6	.657
IBMPQSQACYC	.588	35.8	.627
BBN2002C	.499	28.4	.603
nuslamp2002	.396	21.0	.569
IRST02D1	.589	38.4	.559
isi02	.498	29.8	.555
FDUT11QA1	.434	24.8	.539
ibmsqa02c	.455	29.0	.461
exactanswer	.691	54.2	.449
ilv02wt	.450	30.8	.392
uwmtB3	.512	36.8	.392
ali2002b	.496	36.2	.365
aranea02a	.433	30.4	.357
LCCmain2002	.856	83.0	.168
pris2002	.610	58.0	.095

Table 2: Ranking Ability of Top 15 Submissions

base and inference engine (Cyc) to do sanity checking of answer candidates: the number of times this capability was invoked are too few to do other than say that the approach looks promising. In our future work, we plan to expand PIQUANT's ability to recognize cases when sanity checking is appropriate, improve Cyc's coverage of valid answer ranges, as well as adopt a confidence-based approach to selecting answering agents. Improvements since TREC have led to 16 invocations of the sanity checker on the TREC 2002 question set. These invocations led to one additional correct 1st, 2nd, and 3rd place answers each, validated 4 correct 1st place answers while erroneously validating 3 incorrect 1st place answers, and rejected 122 incorrect answers without any erroneous rejections.

Acknowledgments

We would like to thank Abe Ittycheriah and Salim Roukos for making available their system and results for experimental purposes, Stefano Bertolo for his help with Cyc integration, and Ruchi Kalra for ontology population. This work was supported in part by the Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program under contract number MDA904-01-C-0988.

References

Charles Clarke, Gordon Cormack, and Thomas Lynam. 2001a. Exploiting redundancy in question answering. In *Proceedings of the 24th SIGIR Conference*, pages 358–365.

C.L.A. Clarke, C.V. Cormack, T.R. Lynam, C.M. Li, and McLearn G.L. 2001b. Web reinforced question answering. In *Proceedings of the Tenth Text Retrieval Conference*, pages 673–679.

Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. 2001. The role of lexico-semantic feedback in open-domain textual question-answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 274–281.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. 2000. Question answering in Webclopedia. In *Proceedings of the Ninth Text REtrieval Conference*, pages 655–664.

Abraham Ittycheriah and Salim Roukos. 2002. IBM's statistical question answering system – TREC-11. In *Proceedings of the Eleventh Text Retrieval Conference.*

Abraham Ittycheriah, Martin Franz, and Salim Roukos. 2001. IBM's statistical question answering system – TREC10. In *Proceedings of the Tenth Text Retrieval Conference*, pages 258–264.

Douglas B. Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11).

George Miller. 1995. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11).

Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum, and Vasile Rus. 2000. The structure and performance of an open-domain question answering system. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 563–570.

Marius Pasca and Sanda Harabagiu. 2001. High performance question answering. In *Proceedings of the 24th SIGIR Conference on Research and Development in Information Retrieval*, pages 366–374.

John Prager, Eric Brown, Anni Coden, and Dragomir Radev. 2000. Question-answering by predictive annotation. In *Proceedings of the 23rd SIGIR Conference*, pages 184–191.

John Prager, Dragomir Radev, and Krzysztof Czuba. 2001. Answering what-is questions by virtual annotation. In *Proceedings of Human Language Technologies Conference*, pages 26–30.

John Prager, Jennifer Chu-Carroll, Eric Brown, and Krzysztof Czuba. 2003. Question answering using predictive annotation. In *Advances in Question Answering*. To appear.

Deepak Ravichandran and Eduard Hovy. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 41–47.

Ellen M. Voorhees. 2001. Overview of the TREC 2001 question answering track. In *Proceedings of the 10th Text Retrieval Conference*, pages 42–51.

IBM Research TREC-2002 Video Retrieval System

Bill Adams[*] Arnon Amir[†] Chitra Dorai[‡] Sugata Ghosal[§] Giridharan Iyengar[*]
Alejandro Jaimes[‡] Christian Lang[‡] Ching-yung Lin[‡] Apostol Natsev[‡] Milind Naphade[‡]
Chalapathy Neti[*] Harriet J. Nock[*] Haim H. Permuter[†] Raghavendra Singh[§] John R. Smith[‡]
Savitha Srinivasan[†] Belle L. Tseng[‡] Ashwin T. V.[§] Dongqing Zhang[¶]

Abstract

In this paper, we describe the IBM Research system for analysis, indexing, and retrieval of video, which was applied to the TREC-2002 video retrieval benchmark. The system explores novel methods for fully-automatic content analysis, shot boundary detection, multi-modal feature extraction, statistical modeling for semantic concept detection, and speech recognition and indexing. The system supports querying based on automatically extracted features, models, and speech information. Additional interactive methods for querying include multiple-example and relevance feedback searching, cluster, concept, and storyboard browsing, and iterative fusion based on user-selected aggregation and combination functions. The system was applied to all four of the tasks of the video retrieval benchmark including shot boundary detection, concept detection, concept exchange, and search. We describe the approaches for each of the tasks and discuss some of the results.

1 Introduction

The growing amount of digital video is driving the need for more effective methods for indexing, searching, and retrieving video based on its content. Recent advances in content analysis, feature extraction, and classification are improving capabilities for effectively searching and filtering digital video content. Furthermore, the recent MPEG-7 standard promises to enable interoperable content-based retrieval by providing a rich set of standardized tools for describing features of multimedia content [1]. However, the extraction and use of MPEG-7 descriptions and the creation of usable fully-automatic video indexing and retrieval systems remains a significant technical challenge.

The TREC video retrieval benchmark is facilitating the technical advancement of content-based retrieval of video by standardizing a benchmark video corpus along with different video retrieval and detection tasks. The benchmark provides a consistent evaluation framework for assessing progress as researchers experiment with novel video indexing techniques. This year, we participated in the TREC video retrieval benchmark and submitted results for four tasks: (1) shot boundary detection, (2) concept detection, (3) concept exchange, (4) search. We explored several diverse methods for video analysis, indexing, and retrieval, which included automatic descriptor extraction, statistical modeling, and multi-modal fusion. We conducted experiments that individually explored audio-visual and speech modalities as well as their combination in manual and interactive querying. In the paper, we describe the video indexing and retrieval system and discuss the results on the video retrieval benchmark.

1.1 Outline

The outline is as follows: in Section 2, we describe our process for video and speech indexing. In Section 3, we describe the video retrieval system including methods for content-based search, model-based search, speech-based search, and other methods for interactive searching and browsing. In Section 4, we discuss the approaches for each of the benchmark tasks and examine some of the results.

2 Video indexing system

The video indexing system analyzes the video in an off-line process that involves video content indexing and speech indexing. The video content indexing process consists of shot boundary detection, key-frame extraction, feature extraction, region extraction, concept detection, and clustering, as shown in Figure 1. The basic unit of indexing and retrieval is a video shot.

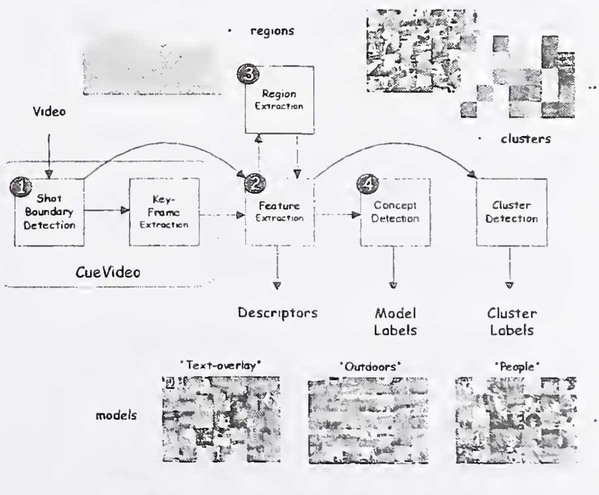


Figure 1: Summary of video content indexing process.

[*] IBM T. J. Watson Research Center, 1101 Kitchawan Rd., Yorktown Heights, NY 10598

[†] IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120

[‡] IBM T. J. Watson Research Center, 19 Skyline Drive, Hawthorne, NY 10532

[§] IBM India Research Lab, Block 1, IIT, New Delhi 110016, India

[¶] Dept. of E.E., Columbia University, New York, NY 10027

2.1 Shot boundary detection (SBD)

Shot boundary detection (SBD) is performed using the real-time *IBM CueVideo* system [2] which automatically detects shots and extracts key-frames. This year, we explored several methods for making SBD more robust to poor video quality. Some of the methods include using localized edge gradient histograms and comparing pairs of frames at greater temporal distances. Overall, our 2002 SBD system showed reduction in errors by more than 30% compared to our 2001 SBD system [3].

The baseline CueVideo SBD system uses sampled. three-dimensional color histograms in RGB color space to compare pairs of frames. Histograms of recent frames are stored in a buffer to allow a comparison between multiple image pairs up to seven frames apart. Statistics of frame differences are computed in a moving window around the processed frame and are used to compute the adaptive thresholds. shown in Figure 2 as a line above the difference measures (Diff1. Diff3 and Edge1). A state machine is used to detect the different events (states). The SBD system does not require any sensitivity-tuning parameters. More details about the baseline system can be found in [3. 4].

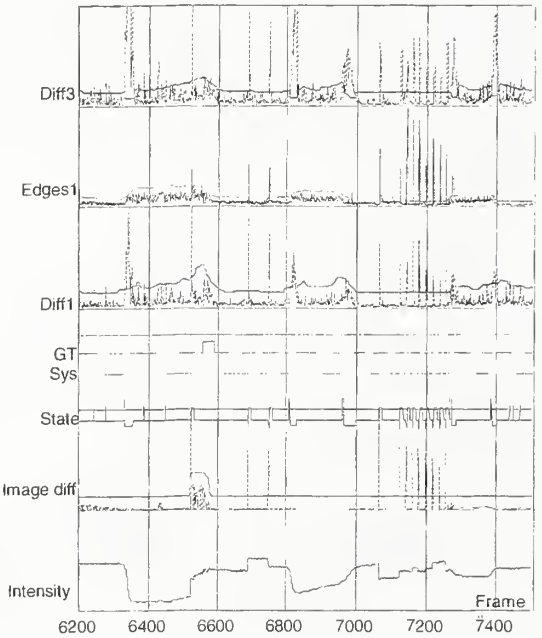


Figure 2: Plot of frame-to-frame processing of the SBD algorithm. Notice the ground truth (GT) and system output (Sys) plots for this segment of video which has six dissolves (one missed) and twelve cuts.

Several changes were incorporated to the baseline SBD algorithm to accommodate lower video quality. as was the case for the videos in the TREC-02 data set. Localized edge-gradient histograms were added to overcome color errors. The 512-bin edge-gradient histogram counts the number of pixels in each of eight image regions, having similar I_x, I_y derivatives (each derivative is quantized into three bits). Thus it is less sensitive to lighting and color changes. Rank filtering was added in time/space/histogram at various different points along the processing to handle the new types and higher levels of noise. The comparison of pairs of frames at wider distances up to thirteen frames apart was added to overcome the high MPEG-1 compression noise. Several new states were added to the state machine to detect certain types of video errors and to detect very short dissolves that were 2-3 frames long. These changes were tuned based on precision-recall measurements using data subsets from TREC01 test set and TREC02 training set.

2.2 Feature extraction

The system extracts a number of descriptors for each video shot. Some of the descriptors, as indicated below. are extracted in multiple ways from each key-frame image using different normalization strategies (see [5]) as follows: (1) global. (2) 4x4 grid, (3) 5-region layout. and (4) automatically extracted regions. The following descriptors were extracted:

- Color histogram (global per key-frame. 4x4 grid. 5-region layout. segmentation regions): one based on a 166-bin HSV color space [6] and another based on 512-bin RGB color space.
- Color correlogram (global per key-frame. 4x4 grid. 5-region layout): based on a single-banded auto-correlogram coefficients extracted for 8 radii depths in 166-color HSV color space [7].
- Edge orientation histogram (global per key-frame. 4x4 grid. 5-region layout): based on Sobel filtered image and quantization to 8 angles and 8 magnitudes [5].
- Wavelet texture (global per key-frame. 4x4 grid. 5-region layout): based on wavelet spatial-frequency energy of 12 bands using quadrature mirror filters [6].
- Tamura texture (global per key-frame. segmentation regions): Three values representing the coarseness. contrast. and directionality, respectively [8].
- Co-occurrence texture (global per key-frame, 4x4 grid. 5-region layout): based on entropy. energy. contrast. and homogeneity features extracted from gray-level co-occurrence matrices at 24 orientations [9].
- Motion vector histogram (global per shot. segmentation regions): based on 8 × 8 motion estimation blocks in the MPEG-1 decoded I and P frames. A six-bin histogram is generated based on the motion vector magnitudes.
- Mel-Frequency Cepstral Coefficients (MFCC): transformation of uncompressed PCM signal to 24 MFCC features including the energy coefficient.

2.3 Region extraction

In order to better extract local features and detect concepts. we developed a video region segmentation system that automatically extracts foreground and background regions from video. The system runs in real-time with extraction of regions from I-frames and P-frames in MPEG-1 video. The segmentation of the background scene regions uses a block-based region growing method based on color histograms. edge histograms, and directionality. The segmentation of the foreground regions uses a spiral searching technique to calculate the motion vectors of I- and P- frames. The motion features are used in region growing in the spatial domain with additional tracking constraints in the time domain. Although we tested MPEG-1 compressed-domain motion vectors, we found them to be too noisy. We also found that combining motion vectors. color, edge, and texture information for extraction of foreground objects did not give significantly better results than using only motion.

2.4 Clustering

We used the extracted visual descriptors (see Section 2.2) to cluster the video shots into perceptually similar groups. We used a k-means clustering algorithm to generate 20 clusters. Wc found color correlograms to achieve an excellent balance between color and texture features. The clusters were later used to facilitate browsing and navigation for interactive retrieval (as described in Section 3.5).

2.5 Concept detection

The concept detection system learns from labeled training video content to classify unknown video content (in our case. the feature test and search test data). We have investigated several different types of statistical models including Support Vector Machines (SVM), Gaussian Mixture Models (GMM) and Hidden Markov Models (HMM).

2.5.1 Lexicon design

The first step in designing a semantic concept detection system is the construction of a concept lexicon [10]. We viewed the training set video and identified the most salient frequently occurring concepts and fixed a lexicon of 106 concepts. which included the 10 concepts belonging to the TREC concept detection task (denoted as primary concepts). Overall, we generated training and validation data and modeled the following 10 primary concepts: Outdoors, Indoors, Cityscape, Landscape, Face, People, Text Overlay, Music. Speech and Monologue. We also modeled the following 39 secondary generic concepts:

- Objects: Person. Road, Building, Bridge. Car. Train. Transportation. Cow, Pig, Dog, Penguin. Fish, Horsc, Animal. Tree. Flower. Flag, Cloud,
- Scenes: Man Made Scenes, Beach. Mountain. Greenery. Sky, Water, Household Setting, Factory Setting, Office Setting, Land. Farm. Farm House. Farm Field, Snow. Desert. Forest, Canyon.
- Events: Parade, Explosion, Picnic, Wedding.

2.5.2 Annotation

In order to generate training and validation data. we manually annotated the video content using two annotation tools[1] – one produced the visual annotations and the other produced audio annotations. The IBM MPEG-7 Video Annotation Tool (*a.k.a. VideoAnnEx*). shown in Figure 3. allows the shots in the video to be annotated using terms from an imported lexicon. The tool is compatible with MPEG-7 in that the lexicons can be imported as MPEG-7 classification schemes and generates MPEG-7 descriptions of the video based on the detected shots and annotations. The tool also allows the users to directly create and edit lexicons.

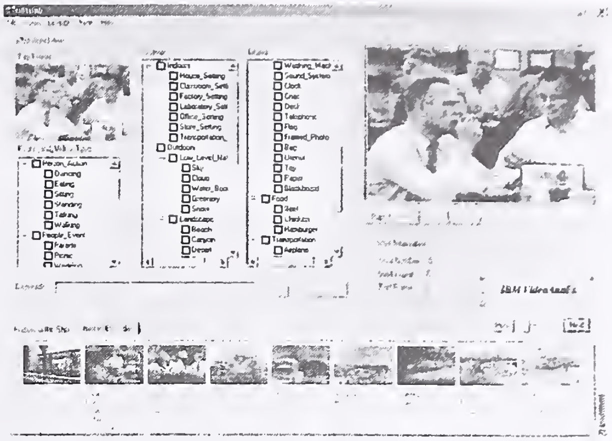


Figure 3: VideoAnnEx MPEG-7 video annotation tool. The system enables semi-automatic annotation of video shots and editing of the lexicon.

The second tool, the IBM Multimodal Annotation Tool, provides three modes of annotation: video. audio with video. or audio without video. The audio annotation is based upon audio segments in which the user manually delimits each segment within the audio upon listening and selects from the lexicon those terms that describe the audio content. Multimodal concepts (e.g. Monologues) are annotated using audio with video mode of annotation.

2.5.3 Concept modeling

Semantic concept detection was investigated using a statistical classification methodology (as described in [11. 12, 10]). The system learns the parameters of the classifiers using training data for each concept using statistical methods. We considered two approaches: one based on a decision theoretic approach and the other based on a risk minimization approach.

Decision theoretic approach In this approach. the descriptors are assumed to be independent identically distributed random variables drawn from known probability distributions with unknown deterministic parameters. For the purpose of classification. we assume that the unknown parameters are distinct under different hypotheses and can be estimated.

Structural risk minimization Unlike the decision theoretic approach. the discriminant approach focuses only on those characteristics of the feature set that discriminate between the two hypotheses of interest. The idea of constructing learning algorithms based on the structural risk minimization inductive principle was proposed in [13]. In particular. we used Support Vector Machines (SVM)[2]. which map the feature vectors into a higher dimensional space through nonlinear function and constructing the optimal separating hyper-plane.

Training and validation Training and validation of models was done using the NIST feature training data set. We randomly partitioned the NIST feature training data set into a 19 hour Feature

[1] Annotation tools are available at http://alphaworks.ibm.com

[2] We used SVMLight toolkit (http://svmlight.joachims.org/)

Training (FTR) collection and a 5 hour Feature Validation (FV) collection. We used the FTR collection to construct the models and the FV collection to select parameters and evaluate the concept detection performance. The validation process was beneficial in helping to avoid over-fitting to the FTR collection.

2.5.4 Fusion

Since no single descriptor is powerful enough to encompass all aspects of video content and separate the concept hypotheses, combining information is needed at several levels in the concept modeling and detection processes. We experimented with two distinct approaches involving early fusion and late fusion. For early fusion we experimented with fusing descriptors prior to classification. For late fusion we experimented with retaining soft decisions and fusing classifiers. In addition, we explored various combining methods and aggregation functions for late fusion of search results as described in Section 3.6. Two modeling procedures are used. They use different subsets of visual features. The first procedure utilizes both early and late fusions, while the second procedure uses only late fusion.

Feature fusion The objective of feature fusion is to combine multiple features at an early stage to construct a single model. However, since this increases the dimensionality of the feature space—which makes it sparser—it also makes the classification problem harder and increases the risk of over-fitting the data. This approach is therefore most suitable for concepts that have sufficiently large number of training set examples that would allow the classifier to exploit correlations between the features. We experimented with feature fusion by simply normalizing and concatenating descriptors. Different combinations of descriptors were used to construct models. We used the validation set to choose the best combination.

Classifier fusion In an ideal situation, early fusion should work for all concepts, since all of the information is available to the classifier. However, practical considerations, such as limited number of training examples and the increased risk of over-fitting necessitate an alternate strategy. If the features are fairly de-correlated, then treating them independently is less of a concern. In such situations, we model concepts in each modality or feature space independently, and fuse individual classifier decisions later. We used a separate model (SVM or GMM) for each descriptor, which results in multiple classifications and associated confidences for each shot depending on the descriptor. While the classifiers can be combined in many ways, we explored normalized ensemble fusion to improve overall classification performance.

2.5.5 Specialized detectors

Although we used the above generic approaches for detection of most concepts, for two concepts (monologues and text overlay) we explored specialized approaches as follows:

Monologue detection For monologue detection, we first performed speech and face detection on each shot. Then, for shots containing speech and face, we further evaluated the synchrony between the face and speech using mutual information and used the combined score thus generated to rank all shots in the corpus. Based on experimental results of a variety of synchrony detection techniques, we used a scheme that models audio and video features as locally Gaussian distributions (see [14] for more details).

Text overlay detection We explored two algorithms for extracted overlay text in video and fused the results of the classifiers to produce the final concept labeling. The first method (see [15]) works by extracting and analyzing regions in a video frame. The processing stages in this system are: (1) isolating regions that may contain text characters, (2) separating each character region from its surroundings and (3) verifying the presence of text by consistency analysis across multiple text blocks. A confidence measure is computed as a function of the number of characters in text objects in the frame. The second method uses macro-block-based texture and motion energy. Layout analysis is used to verify the layout of these character blocks. A text region is identified if the character blocks can be aligned to form sentences or words.

2.6 Speech recognition and indexing

As in TREC-2001, we constructed a speech-based system for video retrieval. Significant improvements were made to both the automatic speech recognition (ASR) performance and the speech search engine performance relative to our TREC-2001 submission.

2.6.1 Automatic speech recognition (ASR)

A series of increasingly accurate speech transcriptions for the entire corpus were produced in the period leading up to the evaluation. The first set of transcriptions were produced using an IBM real-time transcription system tuned for Broadcast News: this is the same transcription system as was used in TREC-2001 [3]. Later transcriptions were produced using an off-line, multiple pass transcription system comprising the following stages (see [16] for more details and citations):

- Remove silent videos
- Divide each video into segments using Bayesian Information Criteria (BIC)
- Detect "music" and "silence" and transcribe using an IBM 10×Real-Time Broadcast News transcription System
- Apply supervised Maximum Likelihood Linear Regression (MLLR) adaptation of speaker-independent HUB4 models using a set of eight (word-level transcribed) videos
- Decode "speech-only" segments using interpolated trigram Language Model (LM)
- Cluster "speech-only" segments into "speaker- and environment- similar" clusters
- Apply unsupervised MLLR adaptation of TREC-2002-adapted HUB4 models to each cluster using single global MLLR mean and precision transforms

The word error rate (WER) of the final transcripts is estimated at 34.6% on a held out set of six videos from Search Test and Feature Test which were manually transcribed[3]. This compares favorably to 39.0% for the best of the publicly-released transcriptions on the same set and represents a 41% improvement over the transcriptions used as the basis for IBM's TREC-2001 SDR system.

[3] Note this set does not overlap with the set used in supervised acoustic model adaptation.

2.6.2 Speech indexing

Indexes were constructed for SDR from the final most accurate speech transcriptions. Three types of indexes were generated: document-level indexes. an inverse word index. and a phonetic index. No attempt was made to index the set of silent videos.

Document-level indexes: The document-level indexes support retrieval at the document level. where a document is defined to span a temporal segment containing at most 100 words[4]. Consecutive documents overlap by 50 words in order to address boundary truncation effects. Once documents are defined and their associated time boundaries are recorded. the documents are pre-processed using (1) tokenization to detect sentence/phrase boundaries; (2) (noisy) part-of-speech tagging such as noun phrase. plural noun etc: (2) morphological analysis. which uses the part-of-speech tag and a morph dictionary to reduce each word to its morph eg. verbs [lands]. [landing] and [land] reduce to /land/; (4) "stop" words are removed using standard stop-word lists. After pre-processing, indexes are constructed and statistics (such as word and word pair term- and inverse-document frequencies) are recorded for use during retrieval.

Inverse word index: the inverse word index supports Boolean search by providing the $(video_i, time_i)$ of all the occurrences of a query term in the videos. Preprocessing of transcripts is similar to that above.

Phonetic index: the phonetic index supports search of out-of-vocabulary words. The (imperfect) speech transcript is converted to a string of phones [17]. The phonetic index can be searched for sound-like phone sequences, corresponding to out-of-vocabulary query terms such as some acronyms. names of people, places. and so forth[5]

3 Video retrieval system

The video retrieval system provides a number of facilities for searching. which include content-based retrieval (CBR). model-based retrieval (MBR). speech-based search or spoken document retrieval (SDR) and other interactive methods.

3.1 Content-based retrieval (CBR)

The objective of CBR is to match example query content to target video content using the extracted descriptors (see Section 2.2). The degree of match is determined on basis of feature similarity. which we have measured using Minkowski-form metrics considering values of $r = 1$ (Manhattan distance) and $r = 2$ (Euclidean distance) as follows: given descriptors represented as multi-dimensional feature vectors, v_q and v_t be the query and target vectors, respectively. then

$$d^r_{q,t} = (\sum_{m=0}^{M-1} |v_q[m] - v_t[m]|^r). \qquad (1)$$

3.2 Model-based retrieval (MBR)

Model-based search allows the user to retrieve video shots based on the concept labels produced by the models (see Section 2.5). In MBR. the user enters the query by typing label text. or the user selects from the label lexicon. Since a confidence score is associated with each automatically assigned label. MBR ranks the shots using a distance $\mathcal{D}$ derived from confidence $\mathcal{C}$ using $\mathcal{D} = 1 - \mathcal{C}$.

3.3 Speech-based search (SDR)

Speech-based search allows the user to retrieve video shots based on the speech transcript associated with the shots. We used multiple SDR systems independently and combined the results to produce the final SDR results for TREC-2002; we refer to the three systems as OKAPI-SYSTEM-1. OKAPI-SYSTEM-2, BOOLEAN-SYSTEM-1. To evaluate different design decisions, a limited ground truth was created for the combined FTR and FV collections by pooling the results and performing relevance assessment.

Query development and preprocessing: All SDR systems operate using a textual statement of information need. Query strings are pre-processed in a similar manner to the documents: tokenization. tagging and morphing gives the final query term sequence for use in retrieval.

Video segment retrieval: Given a query. the three SDR systems rank documents or video segments as follows:

- OKAPI-SYSTEM-1. OKAPI-SYSTEM-2: a single pass approach is used to compute a relevancy score for each document. Each document is ranked against a query. where the relevancy score is given by the OKAPI formula [18]. The total relevancy score for the query string is the combined score of each of the query terms. The scoring function takes into account the number of times each query term occurs in the document and how rare that query term is across the entire corpus. with normalization based upon the length of the document to remove the bias towards longer documents since longer documents are more likely to have more instances of any given word.
- BOOLEAN-SYSTEM-1: a Boolean search was applied to Boolean queries. This search also supported phonetic search of out-of-vocabulary words using the phonetic index. in conjunction with in-vocabulary words which can be located in the inverse word index.

Many SDR systems use the results of first pass retrieval as the basis for automatic query expansion scheme prior to running a second pass of retrieval. Experiments showed little gain from using an LCA-based scheme [19] on FTR+FV. since the number of relevant documents retrieved per query in the first pass is quite low. so the approach was not investigated further.

[4] Minor differences in document definition were used in constructing the different indexes. such as whether or not document boundaries are defined at long stretches of silence or music: experiments suggest these differences do not make a significant contribution to the differences in MAP across systems.

[5] For this year's queries we found the phonetic index was of limited use: only two queries involved out-of-vocabulary words, which were names.

Video segment-to-shot mapping: NIST evaluates video retrieval performance at the level of shots, rather than at the level of documents or video segments which span one or more shots. Thus we must somehow use the scores assigned to documents or video segments by SDR to assign scores at the level of shots[6]. The mappings used in the three component systems are:

- OKAPI-SYSTEM-1: the score assigned to a document is assigned to the longest shot overlapping that document;
- OKAPI-SYSTEM-2: the score assigned to a document is assigned to all the overlapping shots. A slightly higher score given to the later shots than to the first ones:
- BOOLEAN-SYSTEM-1: First, the boundaries of the video segment are determined by the coverage of the relevant words. Then the overlapping shots are scored the same way as with OKAPI-SYSTEM-2.

The video segment-to-shot mapping is critical to overall SDR performance. Post-evaluation experiments show the schemes above were not optimal choices; for example, since multiple relevant shots often overlap a single document, OKAPI-SYSTEM-1 performance can be improved simply by assigning a.document score to all overlapping shots. Our current research is investigating more sophisticated schemes.

Fusion of multiple SDR systems: Analysis of the results from the different systems shows that they are often complementary on FTR+FV: no system consistently outperforms the others. Thus we hypothesized fusion of scores might lead to improved overall performance. Whilst various fusion schemes are possible, for TREC-2002 we use a simple additive weighted scheme to combine shot-level, zero-to-one range normalized scores from each of our basic SDR systems. Weights can be optimized on FTR+FV prior to the final run on (held-out) search test data. This combined system is termed "SDR-FUSION-SYSTEM".

3.4 Term vector search

We used term vectors constructed from the ASR text for allowing similarity search based on textual content. Given the entire collection of shots, we obtained a list of all of the distinct terms that appear in the ASR for the collection. The order of this list was fixed to give a one-to-one mapping of distinct terms and dimensions of the vector space. Each shot was then represented by an n-dimensional vector, where the value at each dimension represented the frequency of the corresponding term in each shot. This allows the comparison of two shots based on frequency of terms. We constructed several term vector representations based on ASR-text.

3.5 Browsing and navigation

The system provides several methods for browsing and navigation. For each video a story-board overview image was generated that allowed its content to be viewed at a glance. The system also generated these overview images for each cluster (see Section 2.4) and each model (see Section 2.5).

3.6 Iterative fusion

The interactive fusion methods provide a way for combining and rescoring results lists through successive search operations using different combination methods and aggregation functions defined as follows:

Combination methods Consider results list R_k for query k and results list Q_r for current user-issued search, then the combination function $R_{i+1} = \mathcal{F}_c(R_i, Q_r)$ combines the results lists by performing set operations on list membership. We explored the following combination methods:

- Intersection: retains only those items present in both results lists.

$$R_{i+1} = R_i \cap Q_r \tag{2}$$

- Union: retains items present in either results list.

$$R_{i+1} = R_i \cup Q_r \tag{3}$$

Aggregation functions Consider scored results list R_k for query k, where $D_k(n)$ gives the score of item with id = n and $Q_d(n)$ the scored result for each item n in the current user-issued search, then the aggregation function re-scores the items using the function $D_{i+1}(n) = \mathcal{F}_a(D_i(n), Q_d(n))$. We explored the following aggregation functions:

- Average: takes the average of scores of prior results list and current user-search. Provides "and" semantics. This can be useful for searches such as "retrieve items that are indoors *and* contain faces."

$$D_{i+1}(n) = \frac{1}{2}(D_i(n) + Q_d(n)) \tag{4}$$

- Minimum: retains lowest score from prior results list and current user-issued search. Provides "or" semantics. This can be useful in searches such as "retrieve items that are outdoors *or* have music."

$$D_{i+1}(n) = \min(D_i(n), Q_d(n)) \tag{5}$$

- Maximum: retains highest score from prior results list and current user-issued search.

$$D_{i+1}(n) = \max(D_i(n), Q_d(n)) \tag{6}$$

- Sum: takes the sum of scores of prior results list and current user-search. Provides "and" semantics.

$$D_{i+1}(n) = D_i(n) + Q_d(n) \tag{7}$$

- Product: takes the product of scores of prior results list and current user-search. Provides "and" semantics and better favors those matches that have low scores compared to "average".

$$D_{i+1}(n) = D_i(n) \times Q_d(n) \tag{8}$$

- A: retains scores from prior results list. This can be useful in conjunction with "intersection" to prune a results list, as in searches such as "retrieve matches of beach scenes but retain only those showing faces."

$$D_{i+1}(n) = D_i(n) \tag{9}$$

- B: retains scores from current user-issued search. This can be useful in searches similar to those above but exchanges the arguments.

$$D_{i+1}(n) = Q_d(n) \tag{10}$$

[6] Whilst this procedure might be simplified by defining documents in a fashion more closely related to shot boundaries, our results to date have found this to be less successful than the approaches discussed above.

3.7 Normalization

The normalization methods provide a user with controls to manipulate the scores of a results list. Given a score $D_k(n)$ for each item with id = n in results set k, the normalization methods produce the score $D_{i+1}(n) = \mathcal{F}_z(D_i(n))$ for each item n as follows:

- Invert: Re-ranks the results list from bottom to top. Provides "not" semantics. This can be useful for searches such as "retrieve matches that are *not* cityscapes."
$$D_{i+1}(n) = 1 - D_i(n) \tag{11}$$
- Studentize: Normalizes the scores around the mean and standard deviation. This can be useful before combining results lists.
$$D_{i+1}(n) = \frac{D_i(n) - \mu_i}{\sigma_i}, \tag{12}$$
where μ_i gives the mean and σ_i the standard deviation, respectively, over the scores $D_i(n)$ for results list i.
- Range normalize: Normalizes the scores within the range $0 \ldots 1$.
$$D_{i+1}(n) = \frac{D_i(n) - \min(D_i(n))}{\max(D_i(n)) - \min(D_i(n))} \tag{13}$$

3.8 Shot expansion

The shot expansion methods allow the user to expand a results list to include for each shot its temporally adjacent neighbors. This can be useful in growing the matched shots to include a larger context surrounding the shots, as in searches such as "retrieve shots that surround those specific shots that depict beach scenes."

3.9 Multi-example search

Multi-example search allows the user to provide or select multiple examples from a results list and issue a query that is executed as a sequence of independent searches using each of the selected items. The user can also select a descriptor for matching and an aggregation function for combining and re-scoring the results from the multiple searches. Consider for each search k of K independent searches the scored result $S_k(n)$ for each item n, then the final scored result $Q_d(n)$ for each item with id = n is obtained using a choice of the following fusion functions:

- Average: Provides "and" semantics. This can be useful in searches such as "retrieve matches similar to item "A" *and* item "B".
$$Q_d(n) = \frac{1}{K}\sum_k (S_k(n)) \tag{14}$$
- Minimum: Provides "or" semantics. This can be useful in searches such as "retrieve items that are similar to item "A" or item "B".
$$Q_d(n) = \min_k(S_k(n)) \tag{15}$$
- Maximum:
$$Q_d(n) = \max_k(S_k(n)) \tag{16}$$
- Sum: Provides "and" semantics.
$$Q_d(n) = \sum_k (S_k(n)) \tag{17}$$
- Product: Provides "and" semantics and better favors those items that have low scoring matches compared to "average".
$$Q_d(n) = \prod_k (S_k(n)) \tag{18}$$

3.10 Relevance feedback search

Relevance feedback based search techniques enhance interactive search and browsing. The user's feedback on a set of shots is used to refine the search and retrieve in minimum number of iterations the desired matches. The user implicitly provides information about the matches being sought or *query concept* by marking whether shots are relevant or non-relevant in relation to his/her desired search output. The system utilizes this feedback to learn and refine an approximation to the user's *query concept* and retrieve more relevant video-clips in the next iteration.

We use a robust relevance feedback algorithm [20] that utilizes non-relevant video-clips to optimally delineate the relevant region from the non-relevant one, thereby ensuring that the relevant region does not contain any non-relevant video-clips. A similarity metric estimated using the relevant video-clips is then used to rank and retrieve database video-clips in the relevant region. The partitioning of the feature space is achieved by using a piecewise linear decision surface that separates the relevant and non-relevant video-clips. Each of the hyper-planes constituting the decision surface is normal to the minimum distance vector from a non-relevant point to the convex hull of the relevant points. With query concepts that can reasonably be captured using an ellipsoid in the feature space. The proposed algorithm gives a significant improvement in precision as compared to simple re-weighting and SVM-based relevance feedback algorithms.

4 Tasks and results

We participated four tasks: shot boundary detection (SBD), concept detection, concept exchange, and search.

4.1 Shot boundary detection (SBD) results

For the shot boundary detection task, the results of five systems were submitted, one of which was last year's SBD system as a baseline. A large difference in performance relative to last year was anticipated due to the degraded video quality of the TREC '02 data. The other four were different versions of the improved system, mainly applying different logic to the fusion of color histogram and the localized edges histogram information. Three of them performed well and yielded very similar results, while the forth one did not perform as well. Table 1 summarizes the evaluation of the baseline system, *alm1*, and the best new system, *sys47*, on last year's and this year's TREC video data test sets. The results on TREC-01 data set were computed by us, while the results for the TREC-02 data set are taken from the official NIST TREC 2002 evaluation of those systems. Two additional rows are provided on TREC-02 benchmark that compare our results to the best and average systems, respectively, among the 54 SBD runs submitted by TREC participants.

As anticipated, the SBD performance on TREC-02 data was lower than on TREC-01 data set. This was very noticeable in other participating systems as well. Never-the-less, the error rates of the

	Video	All		Cuts		Gradual		Frame	
Sys.	Data	Rc	Pr	Rc	Pr	Rc	Pr	Rc	Pr
alml	TR-01	.95	.88	.98	.97	.87	.68	.59	.93
sys47	TR-01	.96	.92	.99	.98	.89	.79	.66	.90
alml	TR-02	.86	.77	.93	.80	.69	.71	.48	.94
sys47	TR-02	.88	.83	.93	.87	.76	.72	.57	.89
S-5	TR-02	.84	.89	.91	.94	.76	.78	.62	.90
mean	TR-02	.76	.79	.86	.84	.53	.60	.55	.71

Table 1: Shot boundary detection results, comparing the new system with last year system on both TREC-01 and TREC-02 video data test sets. If all participating systems are to be ranked by $Pr_{All} + Rc_{All}$ then system *S-5* would be found the best one, provided here for comparison. System *mean* reflects the average of all 54 submitted systems.

new system *sys47* were 20–36% lower than of the baseline system *alml* in almost all measures on both data sets.

4.2 Concept detection results

Overall, concept detection results were submitted for ten concept classes. The evaluation results are plotted in Figure 4, which shows shows Average Precision measured at a fixed number of documents (1000 for the feature test set). The "Average" bars correspond to the performance averaged across all participants. The "Best" bars correspond to the system returning the highest Average Precision. The "IBM" bars correspond to IBM's submitted concept detection run (priority=1). The IBM system performed relatively well on the concept detection task giving highest Average Precision on 6 of the 10 concepts[7].

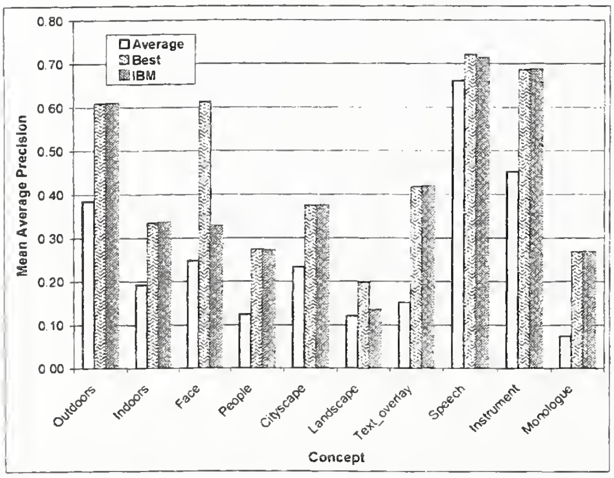


Figure 4: Comparison of concept detection performance using Average Precision.

4.3 Concept exchange results

Apart from running the primary and secondary detectors on the search test set to assist the search task, we participated in the concept exchange task by submitting results of eight primary detectors on the search test set. We generated shot based MPEG-7 descriptions for this exercise thus permitting easy exchange of the detection results between participants.

4.4 Search results

The search task required retrieving video shots from the search test collection for a given set of query topics. We investigated both manual and interactive methods of searching. We submitted four runs of all 25 query topics using the content-based, model-based, speech-based, and interactive search methods described above. Table 2 summarizes the results for the four search runs.

System	Type	Code	MAP
CBR	Manual	M_B_M_1	0.006
SDR	Manual	M_B_M-2_2	0.137
CBR+SDR	Manual	M_B_M-3_3	0.093
CBR+SDR	Interactive	I_B_M-4_4	0.244

Table 2: Summary of search results for four submitted runs.

4.4.1 Manual CBR

The manual CBR run consisted of mapping the query topics into one or more content-based or model-based queries and fusing the results in a predetermined fashion. As described in Section 2.2, CBR was based on a variety of descriptors. The manual CBR run was generated by allowing the following operations to answer each query topic:

1. Issue a content-based search by selecting one or more query examples, a feature type, and a fusion method, as necessary;
2. Issue a model-based search by selecting one or more concept models, a fusion method, and model weights, as necessary;
3. Fuse results lists from one or more content-based or model-based search by selecting a fusion method.

For example, the following sequence of operations was executed for Query 79: *People spending leisurely time at the beach*:

1. Pick examples 0, 1, 4, 8, 12, 23, 29 from query content set
2. Perform CBR search with edge histogram layout using "minimum" fusion (Eq 15)
3. Combine with "Landscape" model using "intersection" combining method (Eq 2) and "product" aggregation function (Eq 8).

The exact mapping of query topics into a fixed sequence of the above operations was performed manually by visually optimizing performance over the FTR and/or FV collections without knowledge of the search test collection. Once a query topic was mapped to system operations, the operations were applied to the search collection by a designated person who did not participate in the mapping process or have prior knowledge of the search test collection. Figure 5 shows the results for topic 76, which is looking for shots depicting "James Chandler." As shown, some matches

[7] Top score is indicated only on five concepts. In our original submission to NIST, we mistakenly submitted the speech detection twice overwriting our instrument detection result. However, the actual Average Precision of our instrument sound detector was 0.686, which was reported through later communication with NIST.

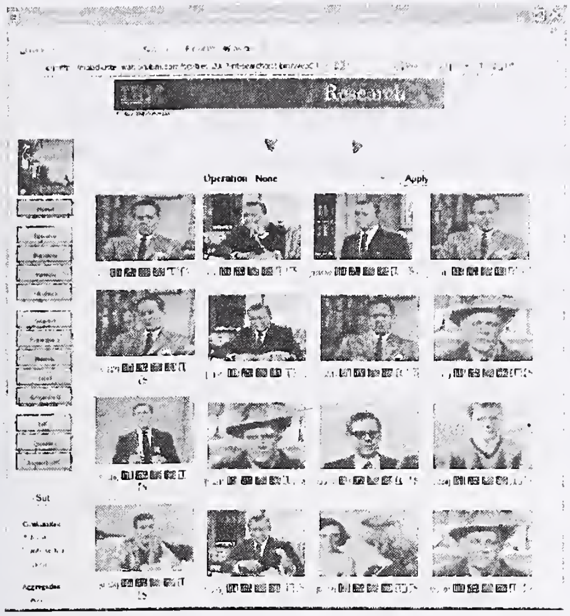


Figure 5: Results for topic 76: James Chandler.

are found in the results list, however, many shots of "James Chandler" are not retrieved using CBR.

With respect to performance, it was our experience that the TREC 2002 query topics were at a higher semantic level than what CBR can handle. While CBR and semantic modeling are generally able to capture low- to mid-level semantics, they are fairly limited in the case of only a few query examples or mid- to high-level semantics. We found that purely CBR worked best for refining candidate lists generated from semantically rich sources, such as speech, or explicit semantic models that closely match the query need. For example, refining the face model by cross-comparison with examples images of "James Chandler" did produce a few relevant hits near the top (see Figure 5). Model-based retrieval on the other hand worked well when the query topic was a close match to an existing model and was built with sufficient training data, such as the "musician" topic. However, in the case of limited example content, such as of query topic looking for "butterflies", or given a lack of closely related explicit semantic models, CBR and MBR techniques alone are not sufficient. In addition, some of the query topics were so general (e.g., beach query) or specific (e.g., Price Tower query) that it is doubtful whether any reasonable discrimination can be done using low-level features alone.

4.4.2 Manual SDR

Manual searching using spoken document retrieval (SDR) was based on the indexed speech information. We explored multiple methods of SDR and their fusion, where the SDR queries were developed through interaction with the Feature Training collection.

Query strings were created manually for each query. Queries derived from the audio and textual statement of information need supplied by NIST were expanded by hand in ad-hoc fashion based on retrieval on the FTR+FV sets[8]. More complicated query strings are used in the Boolean system, since it was hypothesized that the Boolean retrieval would be less susceptible to the effects of query over-tuning on FTR+FV.

The query terms used in the submitted multiple-SDR fusion system for topic 90 ("Find shots with one or more snow-covered mountain peaks or ridges. Some sky must be visible behind them") were "ice snow covered mountain peaks valley vista". Twenty relevant items were retrieved in the top 100, with Average Precision 0.12. For topic 84 ("Find shots of Price Tower, designed by Frank Lloyd Wright and built in Bartlesville, Oklahoma") the query terms are "Price Tower Frank Lloyd Wright Bartlesville Oklahoma", the top three items recalled are relevant and Average Precision is 0.75.

Weights for the SDR-FUSION-SYSTEM were optimized using the limited ground truth that was compiled for FTR+FV. As expected, this scheme led to Mean Average Precision (MAP) improvements FTR+FV; more importantly, fusion gave performance improvements (35%) over our best single SDR system on the unseen search test data (as shown in Table 3). Note that simple post-evaluation changes in the video segment-to-shot mapping scheme improved the performance of the individual OKAPI systems (eg. OKAPI-SYSTEM-1 increased to MAP 0.114) and the fusion system performance might be expected to improve further as the component systems improve. The results overall are a significant improvement over those for IBM's speech-only retrieval submission to TREC-2001. The system was ranked second among 27 evaluated manual search results.

System	MAP
OKAPI-SYSTEM-1	0.073
OKAPI-SYSTEM-2	0.093
BOOLEAN-SYSTEM-1	0.101
SDR-FUSION-SYSTEM	0.137

Table 3: Search test performance of the fusion system and its three components.

4.4.3 Manual CBR and SDR

The combination of CBR and SDR was explored for manual searching, where queries were developed through interaction with the Feature Training collection. An example of (successful) SDR and CBR integration is query topic 86 ("find overhead views of cities - downtown and suburbs; the viewpoint should be higher than the highest building visible"). In the following, we assume that the SDR results and CBR results have been found independently prior to the integrated query:

1. Retrieve results for SDR query of "view panorama overhead downtown suburbs city town urban"
2. Expand results list to include adjacent shots (repeat two times) using expand operation (see Section 3.8)
3. Combine with CBR results using "union" combination method (Eq 3) and "product" aggregation function (Eq 8).

The final Average Precision improved from CBR 0.0 and SDR 0.039 to CBR+SDR 0.057. A similar approach was used for the other queries with minor differences such as the number of shot expansions and the choice of the combination method and aggregation function, for example, using "intersection" rather than "union" and "sum" rather than "product". However, this approach

[8] Later experiments showed that, at least in OKAPI-SYSTEM-1, the gains due to the manual query expansion were negligible.

was not always successful; for example, the same scheme was used for topic 84 ("Price Tower") SDR+CBR but performance was degraded below that obtained using SDR alone. This approach to SDR and CBR integration improved 4 of the 25 queries beyond the performance attained with SDR alone.

4.4.4 Interactive search

We explored interactive search using CBR and SDR in which the user interacted with the search test collection at query-time, we chose various combinations of these methods and selected among different methods for fusion, multiple examples search, relevance feedback, and browsing. The wall-clock time was measured to gauge the user effort for each interactive query. The following describes the interactive search operations for query topic 89 for "Butterflies", which took just over seven minutes of user time:

1. Search for shots of butterflies using SDR with terms such as "monarch", "butterfly", "wings", "flower".
2. View grouping of results by video (clusters shots according to source video) to get idea of which videos contribute which shots
3. Remove two irrelevant shots at top of results list
4. Expand all shots t adjacent shots
5. Results show 5 hits at the top, stop.

5 Summary

We presented the IBM Research video indexing system. The system explores fully-automatic content analysis methods for shot detection, multi-modal feature extraction, statistical modeling for semantic concept detection, and speech recognition and indexing. The system supports manual methods of querying based on automatically extracted features, models, and speech information. In this paper we described the system and the experiments runs that are part of the TREC-2002 video retrieval benchmarking effort. The results show good performance on tasks such as shot boundary detection, concept detection, and search.

Acknowledgments: We thank Prof. Chiou-Ting Hsu, National Tsing-Hua University, Hsinchu, Taiwan and her students for their assistance in annotating the feature training data sets.

References

[1] P. Salembier and J. R. Smith. MPEG-7 multimedia description schemes. *IEEE Trans. Circuits Syst. for Video Technol.*, August 2001.

[2] *IBM CueVideo Toolkit Version 2.1, http://www.almaden.ibm.com/cs/cuevideo/.*

[3] J. R. Smith, S. Srinivasan, A. Amir, S. Basu, G. Iyengar, C.-Y. Lin, M. Naphade, D. Ponceleon, and B. Tseng. Integrating features, models, and semantics for trec video retrieval. In E. M. Voorhees and D. K. Harman, editors, *Proc. Text Retrieval Conference (TREC)*, pages 240–249, Gaithersburg, MD, 2002. NIST.

[4] S. Srinivasan, D. Ponceleon, A. Amir, , and D. Petkovic. What is in that video anyway? in search of better browsing. In *in Proc. of IEEE Intl. Conf. on Multimedia Computing and Systems*, pages 388–392, Florence, Italy, 1999.

[5] J. R. Smith and A. Natsev. Feature and spatial normalization for content-based retrieval. In *IEEE Conference on Multimedia and Expo*, Laussane, Switzerland, August 2002.

[6] J. R. Smith. Content-based access of image and video libraries. In A. Kent, editor, *Encyclopedia of Library and Information Science*. Marcel Dekker, Inc., 2001.

[7] J. Huang, S. Kumar, M. Mitra, W. Zhu, and R. Zabih. Spatial color indexing and applications. *International Journal of Computer Vision*, 35(3):245–268, December 1999.

[8] H. Tamura, S. Mori, and T. Yamawaki. Textural features corresponding to visual perception. *IEEE Trans. Syst., Man, Cybern.*, SMC-8(6):460–473, 1978.

[9] R. Jain, R. Kasturi, and B. Schunck. *Machine Vision*. MIT Press and McGraw-Hill, New York, 1995.

[10] M. Naphade, S. Basu, J. Smith, C. Lin, and B. Tseng. Modeling semnatic concepts to support query by keywords in video. In *IEEE International Confernce on Image Processing*, Rochester, NY, Sep 2002.

[11] M. Naphade, T. Kristjansson, B. Frey, and T. S. Huang. Probabilistic multimedia objects (multijects): A novel approach to indexing and retrieval in multimedia systems. In *Proceedings of IEEE International Conference on Image Processing*, volume 3, pages 536–540, Chicago, IL, Oct. 1998.

[12] M. R. Naphade, I. Kozintsev, and T. S. Huang. A factor graph framework for semantic video indexing. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(1):40–52, Jan 2002.

[13] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[14] G. Iyengar, H. Nock, and C. Neti. Audio-visual synchrony for detection of monologues in video archives. In *Proc. of the Intl. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, April 2003.

[15] J.-C. Shim, C. Dorai, and R. Bolle. Automatic text extraction from video for content-based annotation and retrieval. In *IEEE Conference on Pattern Recognition*, volume 1, pages 618–620, Brisbane, Australia, August 1998.

[16] A. Jaimes, M. Naphade, H. Nock, J. R. Smith, and B. Tseng. Context-enhanced video understanding. In *IS&T/SPIE Symposium on Electronic Imaging: Science and Technology – Storage & Retrieval for Image and Video Databases 2003*, San Jose, CA, January 2003.

[17] A. Amir, A. Efrat, and S. Srinivasan. Advances in phonetic word spotting. In *Proc. of the 2001 ACM CIKM Int. Conference on Information and Knowledge Management*, pages 580–582. ACM, November 2001.

[18] S. E. Robertson, A. Walker, K. Sparck-Jones, M. M. Hancock-Beaulieu, and M. Gatford. OKAPI at TREC-3. In *In Proc. Third Text Retrieval Conference*, 1995.

[19] J. Xu and B. Croft. Improving the Effectiveness of Informational Retrieval with Local Context Analysis. *ACM Transactions on Information Systems*, 2000.

[20] T. V. Ashwin, R. Gupta, and S. Ghosal. Leveraging non-relevant images to enhance image retrieval performance. In *Proc. ACM Intern. Conf. Multimedia (ACMMM)*, Juan Les Pins, France, December 2002.

IIT at TREC-2002

Linear Combinations Based on Document Structure and Varied Stemming for Arabic Retrieval

Information Retrieval Laboratory
Department of Computer Science
Illinois Institute of Technology
Chicago, IL 60616
Abdur Chowdhury, Mohammed Aljlayl, Eric Jensen, Steve Beitzel,
David Grossman, Ophir Frieder
{abdur, aljlayl, ej, steve, grossman, ophir}@ir.iit.edu

Abstract

For TREC 10 we participated in the Named Page Finding Task and the Cross-Lingual Task. In the web track, we explored the use of linear combinations of term collections based on document structure. Our goal was to examine the effects of different term collection statistics based on document structure in respect to known item retrieval. We parsed documents into structural components and built specific term indexes based on that document structure. Each of those indices have their own collection statistics for term weighting based on the type of language used for that structure in the collection. For producing a single ranked list, we examined a weighted linear combination approach to merging results. Our approach to known item retrieval was equal or above the median 58% of the time and 71% above the mean score of submitted runs. In the Arabic track we participated in Arabic Cross-language Information Retrieval (CLIR) and in Arabic monolingual information retrieval. For the monolingual retrieval, we examined the use of two stemming algorithms. The first is a deeper approach, and the second is a pattern-based approach. For the Arabic CLIR, we explored the retrieval effectiveness by using a machine translation (MT) system and translation probabilities obtained from parallel documents collection provided by the United Nations (UN).

Keywords: Known-item search, document structure retrieval, linear combination of retrieval strategies, cross-lingual Arabic retrieval, light-stemming, pattern-based stemming

Named Page Finding Task

Many years of research have been devoted to examining the question of what are the best retrieval strategies for retrieving information, this year we explore a variation on the task where a specific or known-item is sought after given a query or topic. Our research this year specifically explores three basic questions about this task:

How do document structure approaches compare to traditional ranking strategies given that the task and evaluation metrics have changed?

1. What type of document structure can be exploited to improve the effectiveness of this task, in comparison to traditional approaches?

2. How effective are weighted linear combination approaches to combining evidence from document structure retrieval approaches.

Many ranking strategies have been examined in the past. Three of the most studied algorithms are PDLN (Pivoted Document Length Normalization) [1], Okapi BM25 [2], Self-Relevance [3] due to their effectiveness in prior TREC evaluations. In our calibrations, we have found BM25 to perform well so we use it as a baseline.

Some work has already been done on the extraction and storage of HTML term information [4]. Additionally, much has been done with the use of link information to identify hubs and authorities [5]. Since many content developers use HTML elements/tags to improve the readability of their documents, we hypothesize that simply using these tags may improve effectives. There are many different tags that could be used, (e.g.; title, section headers, anchor text, bold, underlines, comments, etc.), but we initially focus on only three types: title, anchor text and text.

Finally, we examine the fusion of different document structure indexes to produce a single ranked list for the know-item task. Those different document representations can be merged with linear combinations maximizing mutual evidence. When combining evidence we extend prior research of weighted linear combination approaches.

In recent years, the category of work known as data fusion or multiple-evidence described a range of techniques in information retrieval whereby multiple pieces of information are combined to achieve improvements in retrieval effectiveness. These pieces of information can take many forms including different query representations, different document representations, and different retrieval strategies used to obtain a measure of relationship between a query and a document. Several researchers have used combinations of different retrieval strategies to varying degrees of success in their systems [6, 7]. Belkin, et al. examined the effects of combining several different query representations to achieve improvements in effectiveness [8, 9]. Lee examined the effect of using different weighting schemes to retrieve different sets of documents using a single query and document representation, and a single retrieval strategy [10].

Fox and Shaw examined combination algorithms that increase the score of a document based on repeated evidence of its relevance, as done in [6]. One of the algorithms designed by Fox and Shaw, CombMNZ, has proven to be a simple, effective method for combining result sets. It was used by Lee in his fusion experiments, and has become the standard by which newly developed result combination algorithms are judged. More recent research in the area of meta-search engines has led to the proposal of several new result combination algorithms of even greater complexity, making use of training data and techniques such as voting algorithms and Bayesian inference [11, 12, 13]. Although these algorithms were shown to behave comparably and occasionally superior to CombMNZ, for our research we use Fox's CombMNZ algorithm, leaving other linear combination approaches as a topic of further research.

In the next section we describe our experimental approach to examine the above questions. In the results section we present our results from this year's experiments. Lastly, we conclude and present future possible research directions

Methodology

To conduct our research we use the IIT retrieval system AIRE [14]. Our system builds a traditional inverted index based on a given document structure(s). Additionally, our system uses conflation classes [15] instead of a more commonly used stemmer such as Porter [16]. Those classes have been modified over the years as problem term variants have been encountered. Additionally, AIRE uses a generated statistical phrase list, where the statistical phrases were generated with a news collection and IDF filtering to reduce the final phrase list size. Phrases are generated from phrases via a bi-gram sliding window algorithm and weighted with 25% importance in relation to keyword weighting for retrieval. Basic term weighting uses the Okapi BM25, Equation 1.

$$\sum \log\left(\frac{(N-n)+.5}{(n+.5)}\right) * \left(\frac{(k1+1)*tf}{(K+tf)} * \frac{(k3+1)*qtf}{(k3+qtf)}\right)$$

$$K = k1*((1-b)+b*dl/avdl)$$

Equation 1: Okapi BM25

Where:

- *tf = frequency of occurrences of the term in the document*
- *qtf = frequency of occurrences of the term in the query*
- *dl = document length*
- *avdl = average document length*
- *N = is the number of documents in the collection*
- *n = is the number of documents containing the word*
- *k1 = 1.2*
- *b = 0.75 or 0.25 (we use .25)*
- *k3 = 7,* set to 7 or 1000, controls the effect of the query term frequency on the weight -- smaller is less.

We indexed the 18GB government collection producing a full-text index, HTML title term index, and an anchor text index. The anchor text index differed from the other indexes, in that an additional mapping stage was required so referencing anchor text data can be linked to the referenced TREC document name. For our experimental layout we first produce a baseline run based on BM25, conflation classes, phrases, full-text index, referred to as the (base) run with the results summarized in Table 4.

Two additional result sets were created; the first one was produced using only the title index and the second produced from only using the anchor text index. With those three indexes and result sets our original three questions can be examined. With a baseline result set, additional document structure techniques can be compared in relation to each other (our first question). Our second question we briefly explore by examining anchor text and title text in relation to full text retrieval. In the next section we present results examining the effectiveness of the various structures and their combinations (our third question) with respect to baseline ad-hoc retrieval strategies.

Our linear combination is a three-step process. First our scores are normalized from each document representation retrieved set using min-max normalization, Equation 3. The advantage of this method is that it preserves all relationships of the data values exactly. It does not introduce any potential bias into the data. Secondly, the final scores are calculated using CombMNZ, Equation 2. Where each individual score is biased via alpha and beta weights assigned to the document structure.

$$\boxed{\text{CombMNZ} = \text{SUM(Individual Similarities)} * \text{Number of Nonzero Similarities}}$$

Equation 2: CombMNZ

$$\boxed{V' = (V - min) * (new\_max - new\_min) / (max - min) + new\_min}$$

Equation 3: Min-Max Normalization

For our linear combination experiments we did not have relevance judgments, thus for our submitted runs we submitted runs based on guesses for the best weighting of linear combinations. Additionally, we limited the combinations of results and weighting to the experiment show in Figure 1.

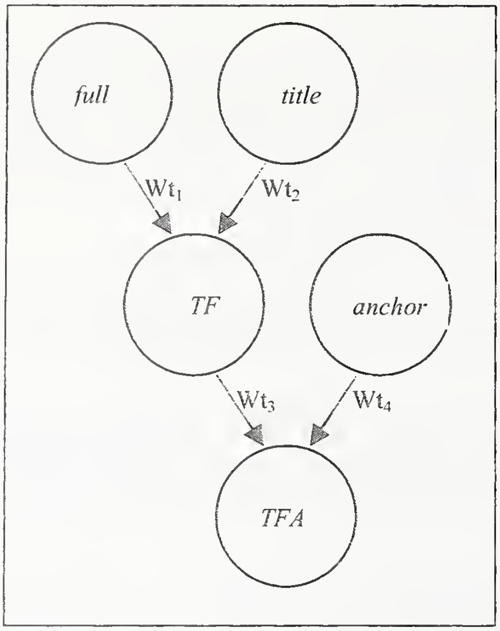


Figure 1: Linear Combination Hierarchy

Results

Our first sets of results examine the retrieval effectiveness of the various document structure elements. In Table 1 we see that the full text index significantly outperforms anchor and title indexes. This is not all that surprising given that anchor retrieval depends on the vocabulary of the referring text, thus not all documents have suitable referring text for the given query set. Similarly, title only retrieval is dependant on the author's vocabulary meeting the query language for this query set and is a subset of the vocabulary used for full text thus performing less than full text is not unexpected.

While the vocabulary for title and anchor text alone may not be as effective as full text for this task/query set, we further examine if combinations of the structures can improve effectiveness. In the second set of experiments we fused the title and full text indices with the CombMNZ algorithm with various alpha, beta weights for each document representation index. Table 2 displays the results of those experiments, while we did not have the final qrels, the run we chose ended up being the best combination of the two retrievals, (alpha=.2 and beta=.8) as highlighted in the table. Effectiveness improvements with any combination of full text and title text retrieval are not found. While a slight improvement is found in the top 10, this does not seem to be a significant improvement of any type.

	Full Text	Anchor	Title
MRR	.587	.156	.323
In Top 10	111	31	67
Found	128	40	82

Table 1: Document Structure Index Runs

MRR	T10	Found	α	β
.402	82	134	.9	.1
.421	90	134	.8	.2
.446	102	134	.7	.3
.468	108	134	.6	.4
.502	109	134	.5	.5
.545	110	134	.4	.6

.559	110	134	.3	.7
.572	***112***	***134***	***.2***	***.8***
.578	111	133	.1	.9

Table 2: Title & Full Fusion, Title = α, Full= β

MRR	T10	Found	α	β
.576	**114**	**134**	**.9**	**.1**
.565	119	134	.8	.2
.539	117	134	.7	.3
.441	115	134	.6	.4
.391	108	134	.5	.5
.297	86	133	.4	.6
.268	68	133	.3	.7
.246	50	133	.2	.8
.218	40	133	.1	.9

Table 3: TF & Anchor Fusion, FT = α, anchor= β

We then fused the combined evidence from (full text and title text) with anchor information and explored various weighting variables. Our submitted run, was the most effective in terms of MRR, but did not yield the greatest number in the top 10. Although the MRR is slightly worse than our full text approach, the number of correct results in the top 10 and "found" increased slightly. After receiving the relevance judgments from NIST we explored other various combination orderings, but found no improvements or negative effects for various orderings of fused results in retrieval effectiveness for all combinations.

These results are surprising given that most popular search engines use document structure for improving the effectiveness of their services. While their improvements may come for other aspects of their approach, using the information as we did showed no significant advantage.

	Base	TF	TFA
MRR	.587	.576	.58
In Top 10	111/74%	114/76%	117/78%
Not Found	22/14.6%	20/13.3%	19/12.6%
>= Mode	82/54.6%	80/53.3%	75/50%
=>Median	88/58.6%	92/61.3%	87/58%
>= Mean	107/71.3%	104/69.3%	108/72%

Table 4: Submitted Result Summary

Our baseline full text retrieval approach for the known-item task was 58% of the time equal or above the median and 71% above the mean score of submitted runs. Additionally, our approach produced the item in the top 10 results 74% of the time and only missed the know-item 14% of the time with 150 queries. Our results using document structure marginally improved top 10 and found statistics, but did not improve MRR. These results are rather surprising in that the BM25 approach had been designed, tuned and tested for a different task and metric. While its success validates the robustness of the algorithm, more research needs to be conducted using document structure to determine how that information should be incorporated into the ranking strategy or that it does not benefit know-item retrieval.

Named-Item Summary

For TREC 10 we explored the use of linear combinations of term collections based on document structure features. Our goal was to examine the effects of different term collection statistics based on document structure in respect to known item retrieval. Our approach is to dissect a document into structural parts and build specific term indexes based on that document structure. Each of those indices would have their own collection statistics for term weighting based on the type of language used for that structure in the collection. For producing a single ranked list, we examined a weighted linear combination approach to merging results.

While our document structure linear combination experiments did not yield any promising results, our approach to known item retrieval was 58% of the time equal or above the median and 71% above the mean score of submitted runs. Additionally, our approach produced the item in the top 10 results 74% of the time and only missed 14% of the known-items out of 150 topics.

Cross-lingual Track

In the Arabic track, we participated in Arabic Cross-language Information Retrieval (CLIR) and in Arabic monolingual information retrieval. We dedicated our effort to improve the retrieval effectiveness of Arabic monolingual retrieval, as we believe it is essential for any Arabic IR or CLIR systems. For the monolingual retrieval, we used two stemming algorithms. The first is a deeper light–based approach, and second is pattern-based approach. For the Arabic CLIR, we explored the retrieval effectiveness by using two recommended standard resources. The resources are a machine translation (MT) system and translation probabilities obtained from parallel documents collection provided by the United Nations (UN). The Arabic AIRE retrieval system is used for experimentation. We used the IIT similarity function and Rocchio relevance feedback.

Background

Unlike alphabets based on the Roman script, the orientation of writing in Arabic is from right-to-left. The shape of most of the characters depends on their position within a word and the character adjacent to them. Most Arabic words are morphologically derived from a list of roots. The root is the bare verb form; it can be triliteral, quadriliteral, or pentaliteral. Most of these roots are made up of three consonants. The Arabic language uses a root-and-pattern morphotactics; patterns can be thought of as templates adhering to well-known rules. These patterns generate nouns and verbs. Roots are interdigitated with the patterns to form Arabic surface forms.

Arabic words are classified into three main parts of speech, nouns (including adjectives and adverbs), verbs, and particles. All verbs and some nouns are derived from a root. Arabic sentences are either verbal or nominal. Verbal sentences contain a verb before the subject, and may contain complements. Nominal sentences begin with a subject followed by a noun, an adjective, a prepositional phrase, or an adverb. In formal writing, Arabic sentences are delimited by commas and periods as in English.

Arabic Monolingual Retrieval

Unlike Indo-European languages such as English, the Arabic language is a highly inflected language. From an Arabic root, many surface forms can be derived. The surface forms of a word have a great impact on a language like Arabic with a strong morphology since surface forms comprise at least two morphemes: a three consonantal root conveying semantic meaning and a word pattern carrying syntactic information. Moreover, most connectors, conjunctions, prepositions, pronouns, and possession forms are attached to the Arabic surface form. Retrieving based on surface form results in low retrieval effectiveness as concluded in [17,18].

Another strategy is to retrieve based on the root of the Arabic word. The goal of the root-based stemmer is to detect and extract the root of an Arabic surface word and it requires very deep syntactic analysis. Al-Shalabi [19] developed a system that detects the root and the pattern of Arabic words with verbal roots. Khoja [20] designed and experimented a novel algorithm for root detection. The retrieval based on the roots improves the retrieval effectiveness as compared to the surface form of the Arabic words. As in our earlier efforts [17,18], light stemming outperforms the root-based stemming. Therefore, light stemming approaches have potential promise [17]

A deeper light stemming approach

The aim of this algorithm is to conflate more related terms in a conflation class than the classes produced in [18]. To achieve this goal, we used a training corpus to identify the frequent suffixes and prefixes. The corpus was obtained from two Saudi Arabian newspapers, namely, Alriyadh and Aljazirah from the year 1999 to 2001. This corpus consists of more than one million words that cover a variety of subjects. The maximum length of the prefixes and suffixes is four letters and the minimum is two letters. Considering more than four letters as prefix of suffix results in ambiguous term after stripping them out. Also, one letter is not enough to form a valid suffix or prefix.

This *automatic* algorithm adheres to the following steps:

1- Check whether the given Arabic word is Arabicized,
2- Remove any diacritics in the given Arabic terms,
3- Start an aggressive normalization,
4- Check for the prefix Waw,
5- Check for duplicate prefixes,
6- Detect definite articles,
7- Check for suffixes,
8- Check for prepositions that attached to the given Arabic stem,
9- Check for prefixes,
10- Normalize the Alf-Maksorah and the Alf.

Throughout the above steps, each step is associated with an event, when the event occurs, an action will be taken. The algorithm checks the length of stem to decide whether to fire the associated action. The minimum length of the stem is three letters. Choosing three letters as minimum maintains the semantic of the Arabic word since most Arabic words are built up from three consonants. In Table 5 we describe some candidate suffixes that considered for removal that we obtained from corpus statistics.

Suffix	Example	Meaning
اتهن	معلماتهن	Their teachers (plural feminine)
اتها	معلماتها	Her teachers (singular feminine)
هما	كتابهما	Their book (dual masculine)
يين	مدنيين	Civilians
تين	تفاحتين	Two apples (dual feminine)

Table 5: Some suffixes derived from the corpus

A pattern-based stemming approach
This approach uses patterns to detect the affixes of the given Arabic word. The algorithm starts first to match a pattern on the given Arabic word. For the case of liberal matching mode, if the matched letters are greater than one, then the algorithm considers that pattern as valid then prefixes and the suffixes will be removed. A more restrictive mode can be applied, i.e., increasing the number of matching between the given Arabic terms and the patterns to consider the current pattern for candidacy. The pattern-based algorithm adheres to the following steps:

1. Remove any diacritics in the given Arabic terms.
2. Normalization such as Alf, and Ya-Maksorah.
3. Check for the prefix Waw
4. Check for duplicate prefixes
5. Detect definite articles
6. Match the given Arabic term on a list of patterns. If there is at least one letter match in the given Arabic term, then the algorithm strips out the suffixes and prefixes of that term based on the matched pattern. If the algorithm fails to extract and remove the suffixes or the prefixes from the given Arabic terms, then the algorithm proceeds executing from step 7 to the end.

To clarify the roles of patterns in Arabic morphology, consider the root (كتب). This root is transliterated as "*ktb* ", which is measured with pattern (فعل). The pattern (فعل) is transliterated as "*fàl*". "*f*" corresponds to the first letter (ـف), *"à"* corresponds to middle letter (ع), and *"l"* corresponds to last letter (ل). The pattern preserves *f*, *à*, and *l* in the same order, whereas vowels and other letters can be added to form a pattern. As shown in Table 6, many patterns are derived from the base pattern *"f à l"* of the root *"ktb"*. As shown, the pattern *"f à alh"* form the word (كتابه) by attaching the vowel (ا) and letter (ه) to the root *"ktb"*. Locating the original letters of the given Arabic word in the pattern is essential step to remove the prefixes and suffixes.

Arabic word	Pattern	Meaning
كاتب	faàl	writer
كتابه	fàalh	writing
الكاتبان	fàl	the two writer
الكاتبين	faàl	The two writers (dual masculine in accusative form)

Table 6: patterns and their surface forms

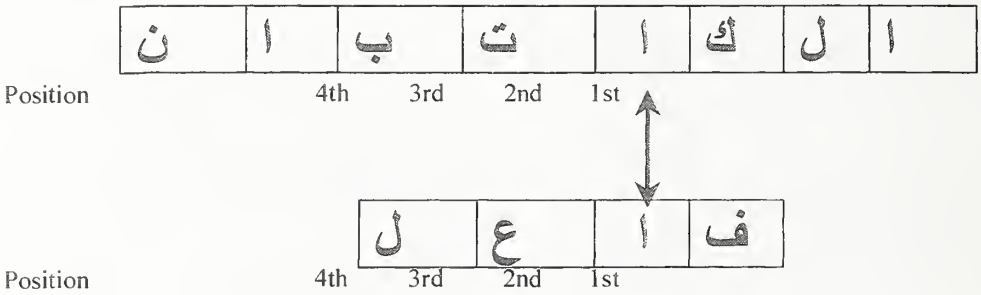


Figure 2. Matching the word "الكاتبان" and the pattern "فاعل"

Figure 2 illustrates the process of matching and stripping out the prefixes and suffixes. Before considering a suffix or prefix for removal, a matching process between the pattern and the given Arabic term is performed. For liberal matching, at least one letter from the pattern should match the Arabic term in same position.

Arabic Cross-language Information Retrieval (CLIR)

In the cross-lingual track, we experimented using the recommended standard resources that are provided by TREC for query translation. We used two means of query translations, machine translation system (MT) and the translation probability that are derived from the UN corpus via BBN [22].

Ajeeb MT system

Machine Translation systems can be defined as any computer-based system that seek automatically to transform a target text from one language into another language by using context information. One of the approaches being used for CLIR is using the existing machine translation system which usually involves automatic translation of the queries, from one language to another. We used ajeeb MT system (www.ajeeb.com) for translating the provided 50 queries (titles and descriptions) from English to Arabic.

Translation probability

Translation probability means that if a term in the source language has several translations in the target language, each term in the target language gets probability. BBN construct translation probabilities that are derived from parallel corpus. The parallel corpus was obtained from the United Nations (UN). The statistical machine translation GIZA++ was used to provide the transaction probabilities. The probability p(a|e) has several terms as candidate for translation, we selected the highest probability for each entry.

Results Analysis

In Arabic monolingual retrieval, our results demonstrate the usefulness of using stemming in improving the retrieval precision. As shown in Table 7, the initial investigation of the pattern-based algorithm, which is in the liberal mode, achieved an improvement over the deeper light stemming algorithm.

	Deeper light stemming	Pattern-based stemming
Average Precision	0.3419	0.3473

Table 7: Average precisions of Deeper light and Pattern-based approaches

In both stemming algorithm, some queries got as close as 0% measured in average precision. The reason behind this drop of retrieval effectiveness is that these queries have fatal error in spelling as well as some has ambiguous term. For example query number 32 got 0% measured in average precision.

صيانة بلوغا في بحرقزوين

The term "بحرقزوين" appears as one single term. As well as, the term "بلوغا" is an ambiguous term. These reasons make our algorithm performed poorly in this query.

Figure 3 demonstrates the average precision at several levels of recalls (0-1) of the pattern-based and deeper light algorithms.

In cross-lingual retrieval, the results of using the translation probabilities performed poorer than machine translation approach as shown in Table 8. The reason behind this drop of retrieval effectiveness is that the construction of the translation probabilities is based on an aggressive stemmer [21] which increases the chance of the co-occurrence of two different terms.

	Machine translation	Statistical translation
Average Precision	0.2453	0.2285

Table 8. Average precisions of Deeper light and Pattern-based approaches

Figure 4 demonstrate the average precision at several levels of recall (0-1), as shown the machine translation system is more effective than the statistical translation.

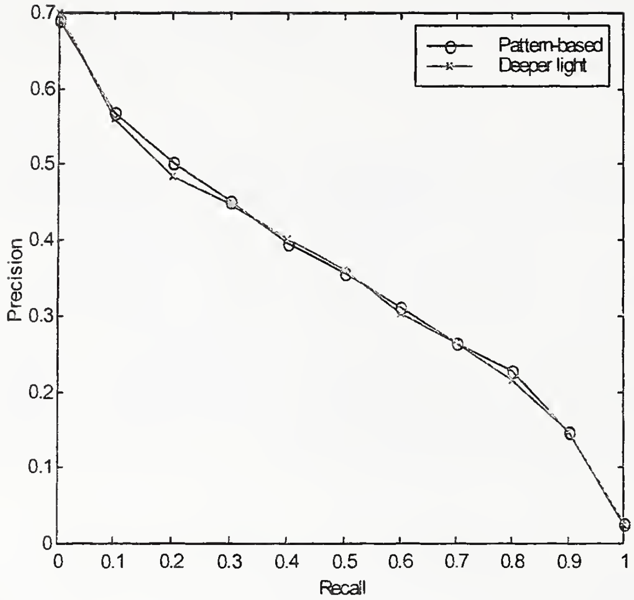


Figure 3. Average precision of pattern-based and deeper light algorithms.

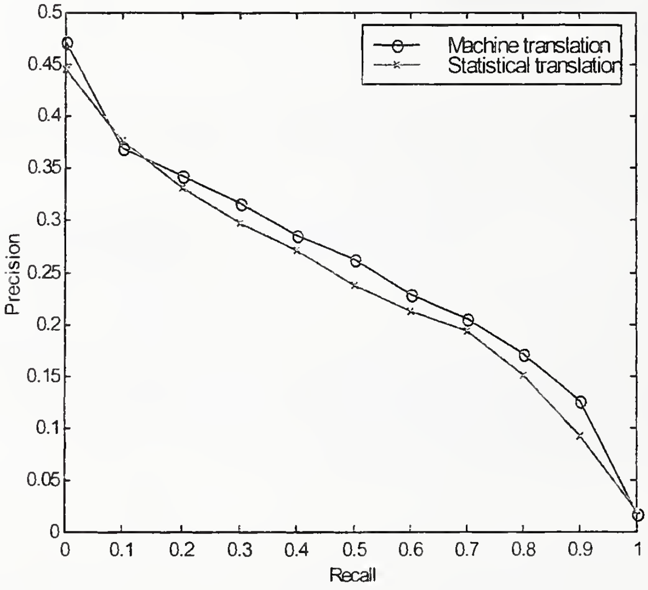


Figure 4. Average precision of machine translation and translations probabilities

CLIR

Summary

We showed that stemming is an important approach to improve the retrieval effectiveness of any Arabic information retrieval systems. In TREC-2002, we participated in both monolingual and cross-lingual retrieval. Our focus in this year is on the improvement of Arabic monolingual information retrieval systems. We presented a new automatic algorithm for stemming, namely, the pattern-based. We experimented with this algorithm by using the liberal mode. In future work, we plan to make it more restricted for matching the given Arabic word and the pattern as well as to increase the number of patterns to enhance the rule-based operations of the algorithm.

In cross-lingual retrieval, we experimented with the standard resources that are provided via TREC11. We found that the machine translation system achieved superior performance compared to translation probabilities. One reason for this is that the construction of the translation probabilities derived from the UN parallel corpus is based on an aggressive stemmer. In addition, some terms are not covered for translation. We plan to enhance the quality of the extracted parallel terms and to add more terms for wider coverage.

References

[1] A. Singhal, et al., "Pivoted document length normalization", ACM-SIGIR, 1996.

[2] S. Robertson, et al., "Okapi at TREC-4", NIST TREC-4, November 1995.

[3] K. Kwok, et al., "TREC-7 Ad-Hoc, High precision and filtering experiments using PIRCS", NIST TREC-7, November 1998.

[4] S. Brin, L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", WWW7 / Computer Networks 30(1-7): 107-117 (1998).

[5] J. Kleinberg, "Authoritative sources in a hyperlinked environment", Journal of the ACM, vol 46, #5, pp 604--632, 1999.

[6] E.A. Fox and J.A. Shaw, "Combination of Multiple Searches," Proceedings of the 2nd Text Retrieval Conference (TREC-2), NIST Special Publication 500-215, pp. 243-252, 1994.

[7] B.T. Bartell. G.W. Cottrell, and R.K. Belew, "Automatic Combination of multiple ranked retrieval systems," Proceedings of the 17th Annual ACM-SIGIR, pp. 173-181, 1994.

[8] N.J. Belkin, C. Cool, W.B. Croft and J.P. Callan, "The effect of multiple query representations on information retrieval performance," Proceedings of the 16th Annual ACM-SIGIR, pp. 339-346, 1993.

[9] N.J. Belkin, P. Kantor, E.A. Fox, and J.A. Shaw, "Combining evidence of multiple query representation for information retrieval," Information Processing & Management, Vol. 31, No. 3, pp. 431-448, 1995.

[10] J.H. Lee, "Combining Multiple Evidence from Different Properties of Weighting Schemes," Proceedings of the 18th Annual ACM-SIGIR, pp. 180-188. 1995.

[11] J. Aslam and M. Montague, et al., "Models for Metasearch", Proceedings of the 24th Annual ACM-SIGIR, 2001.

[12] M. Montague, et al., "Relevance Score Normalization for Metasearch", Proceedings of ACM-CIKM. 2001.

[13] M. Montague, et al., "Condorcet Fusion for Improved Retrieval", Proceedings of ACM-CIKM, November 2002.

[14] A. Chowdhury, et al., "Improved query precision using a unified fusion model", Proceedings of the 9th Text Retrieval Conference (TREC-9), 2000.

[15] J. Xu, B. Croft, “Corpus-based stemming using co-occurrence of word variants”. ACM TOIS, January, 1998.

[16] Porter, “An algorithm for suffix stripping”. Program, 14(3):130—137, 1980.

[17] Aljlayl, M. and Frieder, O. "On Arabic Search: Improving the Retrieval Effectiveness via a Light Stemming Approach”, ACM Eleventh Conference on Information and Knowledge Management, Mclean, VA, November, 2002.

[18] Aljlayl, M., Beitzel, S., Jensen E., Chowdhury, A., Holmes, D., Lee, M., Grossman, D., and Frieder, O."IIT at TREC-10", Proceedings of the Tenth Annual Text Retrieval Conference (TREC-10), NIST, November, 2001.

[19] Al-Shalabi, R, and Evens, M. 1998. A Computational Morphology System for Arabic. Workshop on Computational Approaches to Semitic Languages, COLING -ACL.

[20] Khoja, S. 1999. “Stemming Arabic Text”. Lancaster, U.K., Computing Department, Lancaster University.

[21] TREC 2002 Arabic/English Cross-Language IR Track Resources.

[22] Xu, J., Fraser, A., Weischedel, R. “TREC 2001 Cross-lingual Retrieval at BBN”. Proceedings of the Tenth Annual Text Retrieval Conference (TREC-10), NIST, November. 2001.

VIDEO SEARCHING AND BROWSING USING VIEWFINDER

By

Dan E. Albertson	**Dr. Javed Mostafa**	**John Fieber**
Ph. D. Student	**Associate Professor**	**Ph. D. Candidate**
Information Science	**Information Science**	**Information Science**

School of Library and Information Science
Indiana University

Abstract

Several researchers consisting of students and faculty from the School of Library and Information Science at Indiana University developed a video retrieval system named ViewFinder for the purpose of providing access to video content for a project named the Cultural digital Library Indexing Our Heritage (CLIOH) at Indiana University Purdue University at Indianapolis (IUPUI). For our role in the Text Retrieval Conference (TREC) and its video track, we took the existing system, made notable modifications, and applied it to the video data provided by the conference. After conducting 1 interactive search run, we generated our search results and submitted them to TREC where human judges determined the relevancy of each returned shot and assigned an averaged precision ranking for each topic. From these results we were capable of drawing conclusions of the current system, and how to make ViewFinder more productive in future versions.

Introduction

With the accumulation of digitized video, groups and individuals are becoming more and more interested in the preservation and organization of such content. Along with this preservation and organization, there is a need for systems that can provide easy and efficient access to archived video. This problem is the focus of ViewFinder, a video retrieval information system.

The main goal of Viewfinder is to have it applied to a project being conducted at Indiana University Purdue University at Indianapolis (IUPUI) named the Cultural digital Library Indexing Our Heritage (CLIOH). This project deals with the preservation of multi-media content regarding the ancient world (Mayan ruins, etc.). One such form (of content) is video, and this is the current focus of ViewFinder. ViewFinder attempts to provide users with individual keyframes (of *shots* located within *video files*) according to the user's information need.

For the purpose of participating in the Text Retrieval Conference (TREC) and its video track, we took the existing system, made notable modifications, and applied it to the video data provided by the conference. We then followed conference procedures and performed 1 interactive ("human in the loop") search run consisting of 25 individual topics (also provided by the conference). We then generated results and submitted them to TREC, where human assessors compared our results with the number of manually identified relevant shots, and assigned an average precision for each topic. You may further explore the average precision formula used by TREC in Vorhees, E. M., and Harman, D. K. (2001). In addition to conducting an interactive search run, our system was developed with use and knowledge of the actual search test collection, known as type-A.

Related Literature Review

In recent years there have been various advances in regards to this research problem. This is possibly due to the large increase of multimedia content (especially video) being digitized and made accessible via the World Wide Web and other multimedia information systems.

Along with this increase in video content, there is an increase in people who choose to search for such content. Spink, Goodrum, and Hurson (2001) concluded from a study on Excite query logs between the years 1997 and 1999 that queries for video content increased over 100%. In fact, video queries counted for 0.7% of overall queries in 1997 and counted for 1.6% in 1999 (Spink et al., 2001). Spink et al. (2001) go on to further conclude "video searching became more frequent during this period with the expansion of video material on the Web."

These findings suggest that it is very important for IR researchers to explore how to better provide easier and more efficient access to video content. Cruz and James (1999) provide insight into this problem by focusing on aspects such as user query generation coupled with the user-interface design. They go on to detail their system named Delaunay (Cruz and James, 1999). They express the importance of users having the capability for "pre- and post-query refinement" (Cruz and James, 1999). Furthermore, Cruz and James (1999) also stress the importance of accommodating the search interface to both novice and expert users of multimedia retrieval systems. One example (of their system) is that novice users have the option of a Search Assistant, which may assist in "pre-query refinement" (Cruz and James, 1999).

Spink et al. (2001) also pay close attention to query generation of the user. They claim that, "Web users generally search for multimedia information as they search for textual information" (Spink et al., 2001). Also, Spink et al. (2001), find that multimedia queries contain more search terms (mean of 2.4) than that of general (non-multimedia) Web queries (mean of 1.91).

Spink et al. (2001) further discovered that the term "video" is the most commonly used query term when users search for video content. This brings them to suggest other search features, such as a file extension (.mpeg, .avi, .mov, .wav, etc) search feature, which could be very helpful in user query formation (Spink et al., 2001).

While these search features may prove to be helpful in future (video/image) IR systems, current video/image IR systems still primarily utilize text-based searches. Other research has attempted to increase video/image IR system satisfaction by moving away from total reliance of textual searching and incorporating content-based searching. Zhou and Huang (2002) describe a retrieval system where contextual information (color, shape, texture, etc. described as "low-level features") is combined with user's keywords (or "high-level semantic concepts"). They go on to present that searching on contextual information alone is usually not sufficient in generating relevant results, however, would serve the purpose in thesaurus updating (or adaptation of keywords to images and vice versa) (Zhou and Huang, 2002).

These studies, along with numerous others, reflect the many different directions video retrieval research is currently taking. For example, image and video analysis has been ongoing for many years, and the advancement of this research can be used to explore technologies of (image/video) retrieval. Also, the growing number of users searching for video content has spurred a movement toward applying user-centered design concepts in the development and evaluation of video/image IR systems.

Problem

As mentioned in the earlier sections of this paper, our problem focuses on providing easy and efficient access to video content where large archived (video) data exists. The video data provided by TREC proved to be sufficient in exploring these research problems with ViewFinder. Moreover, the total size of the video collection consisted of 68.45 hours (of MPEG-1) including 40.12 hours for the search test collection, 23.26 hours for the feature development collection, and 5.02 hours for the feature test collection (Smeaton and Over, 2002).

To conduct system tests and the TREC tasks we applied the existing ViewFinder system to the data provided by the conference (through the Internet Archive). Some of the modifications we made to the system consist of a reformulating (system) queries, switch from a MySQL database to Oracle, user-interface adjustments, incorporated a textual keyword search feature, and adapted the search attributes (for proper interaction with the Internet Archive video metadata).

Due to time constraints, the Oracle database resulted in a very basic structure. The indexed metadata for individual *video files* include the titles, descriptions, and descriptors (provided by the Internet Archive), which are identified by an automatically generated video id. For each individual *shot* (keyframe) there is a thumbnail, corresponding URL (to the shot keyframe/thumbnail), and an automatically generated id for both video and shot source. There were a total of 3 tables created for the database. One table contains the shot data (keyframe/thumbnail URL, shot id, and video id), and the other tables contain data corresponding to the video files (e.g. one table contains video id, title, and description fields; and the other table contains video id and descriptor fields).

Although, this proved to be sufficient data to develop a prototype for participating in the video track, we initially assumed that it wouldn't serve the purpose of a practical video retrieval system. Moreover, although TREC evaluates search returns of individual shots (located within a video file), our database only included metadata corresponding to each individual video file. We would encounter the problem of not being capable to distinguish between shots located within the same video file (other than by visually evaluating the shots after a search has been performed).

This prevented us from measuring relevancy rankings between individual shots located within the same video file (e.g. all shots of a matching video are considered "relevant" by the system). For example, once the system identifies any relevant (or matching) video(s), all corresponding keyframes are returned to the user in sequential order. Moreover, if the system matches the user's query to 2 video files, which (both) contain 50 shots, the user would be presented with a shot order such as: shot 1 from video 1, shot 1 from video 2, shot 2 from video 1, shot 2 from video 2 … up until the final shot. (Here, it is up to the user whether or not to browse the returned keyframes.) In the latter sections of this paper we discuss future improvements of ViewFinder, that we feel will eliminate these problems for upcoming TREC conferences.

Methodology

For our interactive ("human in the loop") search run, we allowed the user to evaluate relevancy of the returned results. Moreover, it was up to the user whether or not to reformulate the query and continue searching, or stop and settle on results. In addition, we also made no attempt to restrict or assist the user in query formulation/reformation, nor did we place any time restriction on the user (for individual search topics).

While using the ViewFinder system, the user is given several options of searching techniques (or search features). One of the features consists of a keyword search (See Appendix A for Interface Snapshot). This search allows the user to type in keywords and compare them to the description (field) for each individual video file. If there are any matches between the keyword(s) and any video description, the keyframes corresponding to matching video(s) are returned to the user. Moreover, the keyword search performs a "phrasal" search, or in the case that more than one keyword (a phrase) is entered, the exact phrase must match within the video description in order for results to be returned.

The user is also presented with several (video) attributes in which they are allowed to browse. These attributes are presented to the user in a series of drop down menus (Also See Appendix A for Interface Snapshot). One example is that the user can select "Title" in the "Search By" drop down menu. and retrieve all the video titles in the collection. The user can then select a particular title (by clicking on it and highlighting it) and click the "Search" button, which will run a query for that particular title, and return the associated keyframes.

A similar operation can be conducted with the "Descriptors" option in the drop down search menu. However, unlike the title search (which will only return results for one individual video title) it is possible for the descriptors search to return shots from several different video files (if the same descriptors overlap for multiple videos).

Another search option of ViewFinder is the "Promote" search. This is found in the drop down menus located directly below each of the individual keyframe panels (excluding the middle keyframe). This "Promote" feature will take the descriptors associated with that particular keyframe, and compare it with the descriptors for all other video files, and return any matches. Once the "Promote" search feature has been utilized, the "promoted" keyframe is then displayed in the middle (#5) image panel.

Since ViewFinder can only display up to 9 individual keyframes at one time (8 for search results, and 1 for displaying "promoted" keyframe), the user is still capable of browsing all video shots/keyframes returned (in the case of there being more than 8 matching keyframe). Utilizing the "More Clips" and "Back" buttons located on the interface allows for such browsing. The "More Clips" button becomes initialized after more than 8 keyframes are returned by a query, and the "Back Button" is initialized after the "More Clips" button has been clicked (and the user is on a page other than the first).

These search sessions ended when the user felt they exhausted all relevant video shots. After the user decided to end each of the search topics, they would select the "Finish" button, which would print out up to 100 (top) search results to the Java console, where they were gathered and formatted.

Results

Human assessors from NIST manually judged the relevancy of each returned shot. After concluding on the number of relevant shots returned as compared to the total number of relevant shots identified in the data set, an averaged precision was assigned to each search topic performed. You can read more on the averaged precision formulation in Vorhees, E. M., and Harman, D. K. (2001).

We conducted 1 interactive search run where we attempted to answer all 25 search topics. The mean averaged precision of ViewFinder for all 25 search topics was 0.05472. We had a range of 0.251 with a minimum score of 0.000 (on topics 75 and 85) and a maximum of 0.251 (on topic 76). Ranking among other participating systems included a range from 1st (0.170 topic 94) to a tie for worst (0.000 topics 75 and 85). Moreover, our average ranking for the 25 topics was 17.36 out of an average of 36.88 participating runs. However, there may be some discrepancy in comparing our results with the results of other systems for the reason that search runs (for other systems) varied from interactive to manual, and system development varied from type-A to type-B. (To explore the differences between interactive and manual search runs, and type-A systems and type-B system development please refer to Smeaton, A. F, and Over, P. (2002)).

Conclusions

After reviewing the results, we were initially correct in assuming that the lack of metadata for each individual shot greatly inhibited ViewFinder's searching performance. In future video tracks we plan on populating a database with metadata for each individual shot, which will provide a more robust search for specific information needs. Instead of limiting the search attributes to title, description, and descriptors alone, we would like to add attributes for keywords, subject(s), notable people, important landmarks, and landscapes/cityscapes (just to name a few). Also, we hope to incorporate content-based image retrieval in future versions of ViewFinder. This will allow users to build a more diverse search strategy and allow searches for shots/keyframes with similar shapes, patterns, and colors.

References

Choi, Y. & Rasmussen, E. M. (2002). Users' relevance criteria in image retrieval in American history. *Information Processing & Management, 38*(5), 695-726.

Cruz. I. F., & James, K. M. (1999). User interface for distributed multimedia database querying with mediator supported refinement. *Proceedings of the International Database Engineering and Applications Symposium, Montreal, Canada,* 433 – 441.

Hassan, I, & Zhang, J. (2001). Image search engine feature analysis. *Online Information Review 25* (2), 103 - 114.

Smeaton, A. F., & Over, P. (2002). The TREC 2002 Video Track Report. *Presented at the Eleventh Annual Text Retrieval Conference, Gaithersburg, MD,* November 19th – 22nd, 2002.

Smeaton. A. F., Over, P., & Taban, R. (2001). The TREC-2001 Video Track Report. *NIST Special Publications 500- 250: The Tenth Annual Text Retrieval Conference, Gaithersburg, MD,* 52 – 60.

Spink, A., Goodrum, A., & Hurson, A. R. (2001). Multimedia web queries: Implications for design. *Proceedings of the International Conference on Information Technology: Coding and Computing, Las Vegas, NV,* 589 – 593.

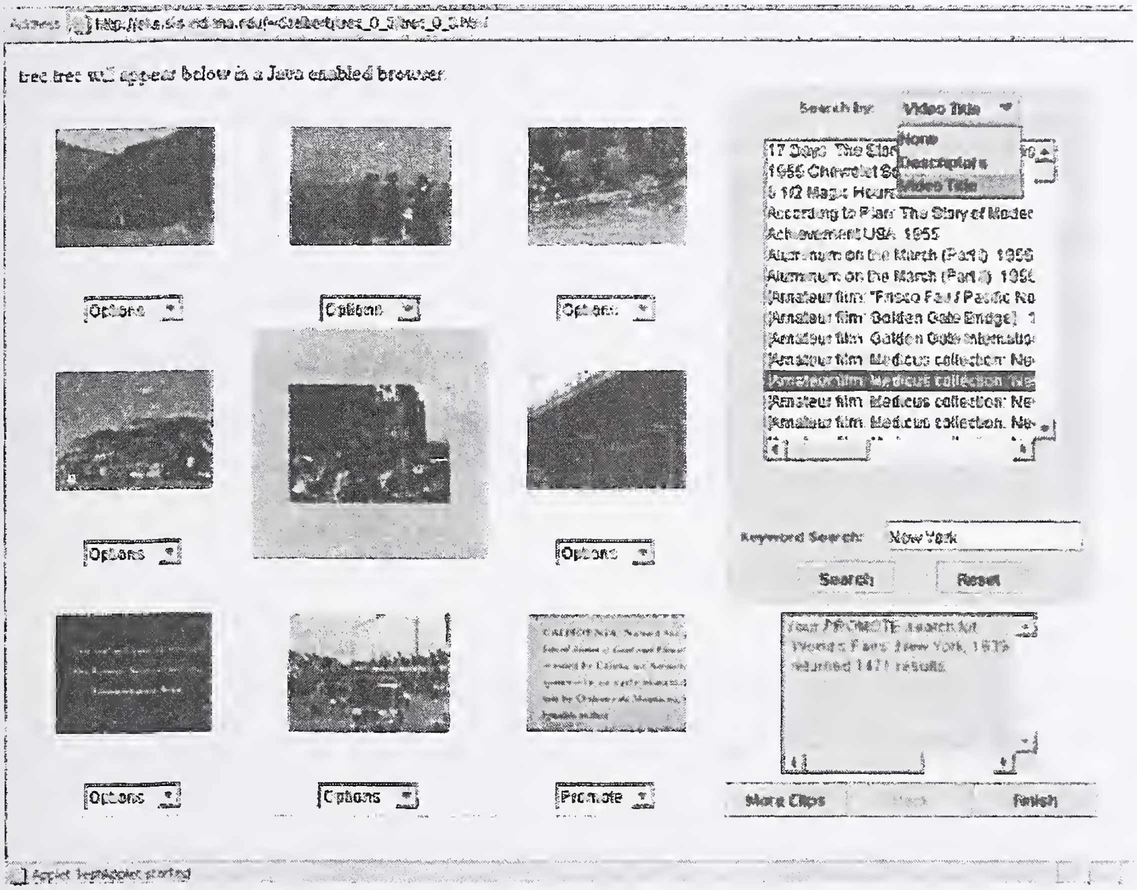
Vorhees, E. M., & Harman, D. K. (Eds.). Common Evaluation Measures. (2001). *NIST Special Publication 500-250: The Tenth Text Retrieval Conference,Gaithersburg, MD*, A14 – A23.

Zhou, X. S., & Huang, T. S. (2002). Unifying keywords and visual contents in image retrieval. *IEEE Multimedia, 9*(2), 23 - 33.

Appendix A

Snapshot of ViewFinder's user-interface. Several search features are being displayed. The "Search By" menu (right hand side) is querying for the titles of the video files. Video titles are then listed in the text box below the "Search By" menu, where one title is highlighted. The "Keyword Search" text field, where the phrase "New York" is entered, is located below the video title listing. Various functions including "Search", "Reset", "More Clips", "Back," and "Finish" buttons are also located below the keyword text field.

The individual thumbnails/keyframes are displayed to the left of the search features. The "Promote" feature has been utilized and the corresponding keyframe is now displayed in the middle image panel.



tree tree will appear below in a Java enabled browser

Search by: Video Title

17 Days: The Stor...
1955 Chevrolet Se...
5 1/2 Magic Hours...
According to Plan: The Story of Moder...
Achievement USA 1955
Aluminum on the March (Part 1) 1956
Aluminum on the March (Part 2) 1956
[Amateur film: Frisco Fair / Pacific No...
[Amateur film: Golden Gate Bridge] 1
[Amateur film: Golden Gate Internatio...
[Amateur film: Medicus collection: Ne...
[Amateur film: Medicus collection: Ne...
[Amateur film: Medicus collection: Ne...
[Amateur film: Medicus collection: Ne...

Keyword Search: New York

Your PROMOTE search for World's Fairs: New York, 1939 returned 1471 results.

Applet TreeApplet started

Video Retrieval using Global Features in Keyframes

Marcus J Pickering[1], Daniel Heesch[1], Robert O'Callaghan[2], Stefan Rüger[1] and David Bull[2]

[1] Department of Computing, Imperial College London,
180 Queen's Gate, LONDON SW7 2BZ, UK
{m.pickering,dh500,srueger}@doc.ic.ac.uk

[2] Image Communications Group, Centre for Communications Research, University of Bristol,
Woodland Road, BRISTOL BS8 1UB, UK
{r.j.ocallaghan,dave.bull}@bristol.ac.uk

Abstract. We describe our experiments for the shot-boundary detection and search tasks for the TREC-11 video track. Our shot-boundary detection scheme is based on a multi-timescale detection algorithm in which colour histogram differences are examined over a range of frames. Our search efforts are based on a system which brings together a number of global features encompassing colour, texture and text features derived from speech recognition transcripts into a unique relevance feedback system.

1 Introduction

Early attempts at content-based video retrieval were based on keywords attached to shots, and this proved very effective for video types such as broadcast news [2, 10]. However, we now live in a multimedia world and, as demonstrated by the search topics for this year's video track, there is potential for querying video in many different ways.

In this paper, we present our system of retrieval of video shots based on global features derived from keyframes. The keyframes are the output of a shot boundary detection process, which we describe in Section 2. Our search system with relevance feedback is described in Section 3.

2 Shot boundary detection task

2.1 System

The video shot boundary detection algorithm is broadly based on the colour histogram method, where the colour histograms for consecutive frames are compared and, if their difference is greater than a given threshold, a shot change is declared. This method is extended, based on the algorithm of Pye et al [12] for detection of gradual transitions that take place over a number of frames, and for rejection of transients, such as the effect of a flash-bulb.

Each frame is divided into 9 blocks, and for each block a histogram is determined for each of the RGB components. The Manhattan distance between corresponding component histograms for corresponding blocks in two frames is calculated, and the largest of the three is taken as the distance for that block. The distance between two frames is then taken as the median of the 9 block distances. This helps eliminate response to local motion.

A difference measure is defined as follows:

$$d_n(t) = \frac{1}{n} \sum_{i=0}^{n-1} D(t+i, t-n+i),$$

where $D(i, j)$ represents the median block distance between frames i and j.

If, at frame f, the value for $d_{16}(f)$ is greater than an empirically determined threshold T_{16}, the frame is examined for the presence of a shot change.

A cut is declared at frame f if the following conditions hold:

- $d_8(f) > T_8$ (where $T_8 = 0.4T_{16}$)
- $d_n(f) > d_n(f + \delta)$ for all $n \in \{2, 4, 8\}$ and all $\delta \in \{-2, -1, 1, 2\}$ (cuts show characteristic coincident peaks for all d_n).
- $d_n > 1.3d_{n/2}$

If no cut was declared, a gradual transition is declared if the following conditions hold:

- $d_{16}(f) > d_{16}(f \pm \delta)$ for all $\delta \leq 16$
- Peak value of d_8 in range $f \pm 16$ occurs within $f \pm 5$

In order to determine the start and end points for gradual transitions, we employ a method similar to that described by Zhang [19], in which a lower threshold, T_4, is used to test for the start and end of a gradual transition. At each frame, the d_4 difference is compared to the threshold. If $d_4 < T_4$ then the frame is marked as a potential start of a transition. If, on examination of successive frames, d_4 falls below T_4 again before a shot change is detected, this potential start is scrapped and the search continues. Following the detection of a shot change, the end point of the transition is declared as the point at which d_4 first falls below the threshold again, following the shot change. The d_4 timescale is used because it is fine enough to pinpoint accurately the moment at which the change begins, but also introduces a tolerance to any momentary drop in the difference which may occur in the process of the change.

It has been suggested that automatic threshold setting can improve performance [13], but we found no empirical evidence to support this with our algorithm. We were, however, able to improve on our TREC-10 system [11] by using empirical data to determine the cut and gradual definition rules.

2.2 Experiments

We performed six shot boundary detection runs. The first three runs, KM-01 – KM-03 were carried out keeping the low threshold, T_4, constant, and reducing the high threshold, T_{16}. In runs KM-04 – KM-06, the low threshold was increased and the same 3 values for the high threshold were used again.

2.3 Results

	All		Cuts		Gradual			
	Recall	Prec	Recall	Prec	Recall	Prec	F-Recall	F-Prec
KM-01	0.826	0.843	0.883	0.895	0.682	0.707	0.673	0.608
KM-02	0.845	0.798	0.889	0.863	0.733	0.648	0.658	0.618
KM-03	0.859	0.720	0.893	0.803	0.773	0.553	0.650	0.612
KM-04	0.825	0.813	0.888	0.880	0.665	0.645	0.471	0.603
KM-05	0.833	0.755	0.891	0.832	0.685	0.578	0.477	0.444
KM-06	0.836	0.688	0.885	0.755	0.711	0.536	0.477	0.356
TREC Avg	0.760	0.790	0.852	0.835	0.527	0.603	0.551	0.713

Table 1. Shot boundary detection task – results summary

We show the results for our six shot-boundary detection runs in Table 1. All six runs gave good results for overall precision and recall, comparing favourably with the average of all systems (shown as "TREC Avg" in Table 1). System KM-01 appeared to give the best balance between precision and recall overall, suggesting that further experiments with a higher T_{16} threshold may be worthwhile.

The frame-recall and frame-precision results (F-Recall and F-Prec respectively in Table 1) give an indication of the accuracy of the system for detection of gradual transitions. Our relative performance here was not as good, and this perhaps reflects the fact that little time was devoted to tuning the algorithm for setting the start and end times of gradual transitions. Results could almost certainly be improved here by adjusting the parameters of this algorithm.

3 Search task

3.1 Overview

For each shot of each video, we take a representative *keyframe*, defined as the middle frame of the shot. The shot boundaries were prescribed by NIST for the task. For each keyframe, a number of feature vectors are pre-computed. The descriptors are then combined in an integrated retrieval model such that the overall distance between a query set $\hat{Q}$ and an image T is given by a convex combination of the distance values computed for each descriptor.

$$D(\hat{Q},T) = \sum_{d} w_d \mathrm{Dis}_d(\hat{Q},T) \tag{1}$$

where $\mathrm{Dis}_d(\hat{Q},T)$ denotes the distance for descriptor d between query set $\hat{Q}$ and T, $w_d \in [0,1]$ and $\sum_d w_d = 1$. The descriptor-specific distance values are computed using the k-nn method, described in Section 3.3.

We combined 6 features, described in the next section. Following retrieval, the user has the option to apply relevance feedback, through a system described in section 3.4.

3.2 Features

HSV Colour Histograms. Retrieval from image databases using only colour was one of the earliest content-based retrieval methods [4, 9, 14]. There is an abundance of colour spaces [5, 16, 18], virtually all of which are 3-dimensional owing to the human perception of light using three different cones as receptors in the retina. Colour histograms are quantised distributions in the 3-dimensional colour space of all pixels of one image. The corresponding feature vector is a list of the proportions of pixels which fall into the respective 3-dimensional colour bins; its length depends on the granularity of the colour bins. Here, we do *not* use 1-dimensional component-wise histograms since (as with all marginalisations) information about the underlying colours would be lost.

HSV [16] seems to be intuitive to humans. The hue coordinate H encodes the underlying pure colour tone of a colour circle. The saturation S reflects the pureness of the colour (the less pure the colour the more grey is mixed into it, S is zero for greys). V and L are both measures, albeit differently defined, for the apparent brightness or luminosity. When expressing the difference of two colours humans tend to use HSL or HSV coordinates ("more in direction of magenta", "purer than", "darker than") rather than RGB components.

HSV and HSL are both cylindrical colour spaces with H being the angular, S the radial and V or L the height component. This brings about the mathematical disadvantage that hue is discontinuous wrt RGB coordinates and that hue is singular at the achromatic axis $r = g = b$ or $s = 0$. As a consequence we merge, for each brightness subdivision separately, all pie-shaped 3-d HSV bins which contain or border $s = 0$. The merged cylindrical bins around the achromatic axis describe the grey values which appear in a colour image and take care of the hue singularity at $s = 0$. Saturation is essentially singular at the black point in the HSV model and at both black and white points in the HSL model. Hence, a small RGB ball around black should be mapped into the bin corresponding to $hsv = hsl = (0,0,0)$, or $hsl = (0,0,1)$ respectively for white, to avoid jumps in the saturation from 0 to its maximum of 1 when varying the singular RGB point infinitesimally. There are several possibilities for a natural subdivision of the hue, saturation and brightness axes; they can be i) subdivided linearly, ii) so that the geometric volumes are constant in the cylinder and iii) so that the volumes of the nonlinear transformed RGB colour space are nearly constant. The latter refers to the property that few RGB pixels map onto a small dark V band but many more to a bright V interval of the same size; this is sometimes called the HSV cone in the literature. We use the HSV model with a linear subdivision.

Convolution filters. For this feature we use Tieu and Viola's method [15], which depends on the definition of highly selective features that are determined by the structure of the image, as well as capturing information about colour, texture and edges. By defining a vast set of features, each feature is such that it will only have a high value for a small proportion of images, and by discovering a number of features which distinguish the example set in question we are able to perform an effective search.

The feature generation process is based on a set of 25 primitive filters, which are applied to each of the three colour channels to generate 75 feature maps. Each of these feature maps is rectified and downsampled before being fed again to each of the 25 filters to give 1875 feature maps. The process is repeated a third time, and then each feature map is summed to give 46,875 feature values. The idea behind the three stage process is that each level 'discovers' arrangements of features in the previous level. The feature generation process is computationally quite costly, but only needs to be done once and then the feature values can be stored with the image in the database.

Text. Our text feature is derived from the speech recognition transcripts supplied by Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur (LIMSI). A full text index was built using the Managing Gigabytes search engine [1] and queries formed from the XML data supplied with each query. Managing Gigabytes supplies a numerical relevance value which is used when weighing features.

HMMD Colour Histogram. The recently introduced HMMD (Hue, Min, Max, Diff) colour space, which is used in the MPEG-7 standard, is derived from the HSV and RGB spaces. The hue component is the same as in the HSV space, and max and min denote the maximum and minimum among the R, G, and B values, respectively. The diff component is defined as the difference between max and min. Three components are sufficient to uniquely locate a point in the colour space and thus the space is effectively three-dimensional. Following the MPEG-7 standard, we quantize the HMMD non-uniformly into 184 bins with the three dimensions being Hue, Sum and Diff (sum being defined as $(max + min)/2$) and use a global histogram. See Manjunath and Ohm [6] for details about quantization.

Colour Structure Descriptor. We use a second descriptor defined in HMMD space that lends itself better to capturing local image structure. A 8×8 structuring window is used to slide over the image. Each of the 184 bins of the HMMD histogram contains the number of window positions for which there is at least one pixel falling into the bin under consideration. This descriptor is capable of discriminating between images that have the same global colour distribution but different local colour structures. Although the number of samples in the 8×8 structuring window is kept constant (64), the spatial extent of the window differs depending on the size of the image. Thus, for larger images appropriate sub-sampling is employed to keep the total number of samples per image roughly constant. The bin values are normalized by dividing by the number of locations of the structuring window and fall in the range $[0, 1]$ (see Manjunath and Ohm [6] for details).

Illumination Invariant Colour Descriptors. Recognition and retrieval via colour are heavily influenced by variations in the scene illumination conditions. This places undesirable limitations on the use of raw colour features in content-based applications. In an attempt to attain some robustness to variation in lighting conditions, we use the set of illumination-invariant descriptors defined by O'Callaghan and Bull [8]. These are histogram, rather than pixel, based features and are calculated using invariant moments of the distribution in RGB space. In the current implementation, they are applied on a global basis to each key-frame. As such, they provide a small number of features (specifically 21), which describe the colour distribution of the scene, invariant to changes in the colour and intensity of the illuminant. Spatial variation of the illumination over the scene is neglected and a diagonal model of illumination change is assumed.

The utility of these colour descriptors was previously demonstrated by O'Callaghan and Bull [8] on a constrained dataset [3], of images of colourful man-made objects, under varying illumination. One of the objectives of our search task submission was to evaluate the performance of such features in a "real" retrieval application in comparison with conventional methods.

3.3 Retrieval using k-nearest neighbours

Retrieval is performed using the k-nearest neighbour approach, which is based on the intuitive notion that if we have seen and already identified something, then anything we later see that is the same, or similar, (based on some defined characteristics) is probably the same kind of thing. So, we provide positively and negatively classified examples, and then classify all test images according to their proximity to the examples.

We use a variant of the distance-weighted k-nearest neighbour approach [7]. Positive examples are supplied by the user, and a number of negative examples are randomly selected from the database. The distances, for descriptor d, from the test image T_i to each of the k nearest positive or negative examples (where 'nearest' is defined by the Euclidean distance in feature space) are determined, and a distance measure calculated as follows:

$$\mathrm{Dis}_d(\hat{Q}, T_i) = \frac{\sum_{n \in N} (\mathrm{dist}(T_i, n) + \varepsilon)^{-1}}{\sum_{q \in \hat{Q}} (\mathrm{dist}(T_i, q) + \varepsilon)^{-1} + \varepsilon}$$

where $\hat{Q}$ and N are the sets of positive and negative examples respectively amongst the k nearest neighbours, such that $|\hat{Q}| + |N| = k$. ε is a small positive number to avoid division by zero. Images are ranked according to $\mathrm{Dis}_d(\hat{Q}, T_i)$.

3.4 Relevance feedback

Retrieved images $T_1, T_2, \ldots$ are displayed as thumbnails such that their respective distance from the centre of the screen is proportional to the dissimilarity $D_s(\hat{Q}, T_i)$ (given by Equation 1) of thumbnail T_i to the query set $\hat{Q}$. Using this semantics of thumbnail location on the screen, the user can provide relevance feedback by moving thumbnails closer to the centre (meaning they are more relevant than the system predicted) or further away (indicating less relevance). The user effectively supplies the system with a real vector of distances $D_u(\hat{Q}, T_i)$, which, in general, differ from the distances $D_s(\hat{Q}, T_i)$ which the system computes using the set of weights w_d. The sum of squared errors

$$\begin{aligned} \mathrm{SSE}(w) &= \sum_{i=1}^{N} \left[D_s(\hat{Q}, T_i) - D_u(\hat{Q}, T_i) \right]^2 \\ &= \sum_{i=1}^{N} \left[\sum_{d} w_d \mathrm{Dis}_d(\hat{Q}, T_i) - D_u(\hat{Q}, T_i) \right]^2 \end{aligned} \tag{2}$$

gives rise to an optimisation problem for the weights w_d such that (2) is minimised under the constraint of convexity. Using one Lagrangian multiplier we arrive at an analytical solution w' for the weight set which changes the distance function. We get a different ranking of images in the database and, with (1), a new layout for the new set of top-retrieved images on the screen.

3.5 Experiments

We carried out four runs to investigate the effects of various combinations of features and of relevance feedback:

1. All features + using relevance feedback.
2. Illumination invariant, Text and Convolution features only.
3. All features. (Baseline for run 1).
4. CSD, Text and Convolution features only. (Baseline for run 2).

3.6 Results

Topic	I_B_KM-1_1	M_B_KM-2_2	M_B_KM-3_3	M_B_KM-4_4
75	0.172	0.146	0.142	0.146
76	0.487	0.540	0.545	0.442
78	0.000	0.188	0.000	0.172
80	0.081	0.009	0.146	0.071
81	0.138	0.000	0.000	0.000
83	0.133	0.028	0.000	0.024
84	0.260	0.250	0.050	0.258
92	0.121	0.021	0.011	0.033

Table 2. Search task – results for topics for which at least one variant of our system achieved average precision greater than 0.100

In Table 3.6 we show the results for topics for which our system achieved average precision greater than 0.100. The topics for which we achieved our best results are, at first glance, surprising – topics 75 and 76 were both queries requiring specific personalities, Eddie Rickenbacker and James H Chandler. Our system was not designed to detect faces. However, both queries contained film of quite distinctive colouring, and the Chandler query contained query shots from within the test set.

Using our four runs we hoped to show that relevance feedback improved performance and that the use of illumination invariant features improved performance, but results were not completely conclusive for either hypothesis.

As we carried out our interactive (relevance feedback) run (I_B_KM-1_1) the retrieved shots were certainly visually much better with each round of relevance feedback, though this is not spectacularly clear from the numerical results. There was some improvement in the average precision for most topics, an observation which is reinforced by the 95% confidence interval for the difference between the performance means of the results for this run and its baseline (M_B_KM-3_3) which, while not proving statistical significance, does suggest an improvement in performance using relevance feedback. The perceived performance improvement may simply be due to the fact that relevance feedback re-ordered the rankings – and with better top-ranked results the user's overall impression is one of greater satisfaction. Some topics (for example 81 – football players, 83 – Golden Gate Bridge, 88 – US maps) benefitted significantly from the application of relevance feedback.The interactive runs in TREC model a search scenario where someone, such as a librarian, searches on behalf of someone else who ultimately judges the returned results. This is different from our model of relevance feedback where the searcher is the one who judges and uses the results. It is also important to note that a user may often be more content with one or two good results, highly ranked, than with retrieving every relevant item in the database.

Calculation of the 95% confidence interval for the difference between the means of the results of the illumination invariant run (M_B_KM-2_2) and its baseline (M_B_KM-4_4) showed that the introduction of the illumination invariant feature brought about no overall improvement in results. However, performance was improved in a number of specific topics, for example, topics 90 – snow covered mountains, and 91 – parrot.

With hindsight, the experiments could have been better designed; in some cases the limited number of features used in the second run (run M_B_KM-2_2) performed better than the combination of all features (run M_B_KM-3_3), suggesting that a philosophy of "more features is better" does not necessarily hold. Some further experiments could be carried out to discover which combinations of features work best – whether there are some features that are consistently good and some that are consistently unhelpful, and whether some features facilitate good results in the presence or absence of other particular features.

4 Conclusions

Our shot boundary detection scheme was shown to work very effectively, with particularly good performance on cuts. There is some potential for improvement of its accuracy on gradual transitions, though detection was good and well above average.

In the search task, we have shown that the use of global features in keyframes, with the addition of relevance feedback, can make an effective contribution to retrieval. Although the recall levels were not spectacular, subjectively the top-ranked results were good on many of the topics.

Further experiments are required to determine which are the best combinations of features, and which features contribute significantly (positively or negatively) to the retrieved results.

Acknowledgements: This work was partially supported by the EPSRC, UK.

References

1. Managing Gigabytes search engine. http://www.cs.mu.oz.au/mg/.
2. M. G. Brown, J. T. Foote, G. J. F. Jones, K. Spärck-Jones, and S. J. Young. Automatic content-based retrieval of broadcast news. In *Proceedings of the Third ACM Multimedia Conference*, Apr. 1995.
3. Computational Vision Lab - Simon Fraser University. Data for computer vision and computational colour science. Available: http://www.cs.sfu.ca/~colour/data/.
4. C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4):231–262, 1994.
5. R. Jackson, L. MacDonald, and K. Freeman. *Computer Generated Colour.* Wiley, 1994.
6. B. S. Manjunath and J.-S. Ohm. Color and texture descriptors. *IEEE Transactions on circuits and systems for video technology*, 11:703–715, 2001.
7. T. M. Mitchell. *Machine Learning.* McGraw Hill, 1997.
8. R. J. O'Callaghan and D. R. Bull. Improved illumination-invariant descriptors for robust colour object recognition. In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, 2002.
9. A. Pentland, R. Picard, and S. Sclaroff. Photobook: Content-based manipulation of image databases, 1994.
10. M. J. Pickering. Video archiving and retrieval. http://km.doc.ic.ac.uk/video-se/, 2000.
11. M. J. Pickering and S. M. Rüger. Multi-timescale video shot-change detection. In Voorhees and Harman [17].
12. D. Pye, N. J. Hollinghurst, T. J. Mills, and K. R. Wood. Audio-visual segmentation for content-based retrieval. In *5th International Conference on Spoken Language Processing, Sydney, Australia*, Dec. 1998.
13. J. R. Smith, S. Srinivasan, A. Amir, S. Basu, G. Iyengar, C.-Y. Lin, M. Naphade, D. Ponceleon, and B. Tseng. Integrating features, models, and semantics for TREC video retrieval. In Voorhees and Harman [17].
14. M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, 1991.
15. K. Tieu and P. Viola. Boosting image retrieval. In *5th International Conference on Spoken Language Processing*, Dec. 2000.
16. D. Travis. *Effective Color Display.* Academic Press, San Diego, CA, 1991.
17. E. M. Voorhees and D. Harman, editors. *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, 2001.
18. G. Wyszecki and W. S. Stiles. *Color Science: Concepts and Methods, Quantitative Data and Formulas.* Wiley, 2nd edition, 1982.
19. H. J. Zhang, A. Kankanhalli, and S. W. Smoliar. Automatic partitioning of full-motion video. *ACM Multimedia Systems*, 1:10–28, 1993.

Use of Patterns for Detection of Answer Strings: A Systematic Approach

Martin M. Soubbotin, Sergei M. Soubbotin
InsightSoft-M, Moscow, Russia
http://insight.com.ru/

Abstract

The paper describes the Question Answering approach applied first at TREC-10 QA track and developed systematically in TREC 2002 experiments. The approach is based on the assumption that answers can be identified by their correspondence to formulas describing the structure of strings carrying certain (generalized) semantics, supposed by the question type. These formulas, or patterns, are like regular expressions but include elements corresponding to predefined lists of terms. Complex patterns can be constructed from blocks corresponding to such semantic entities as persons' or organizations' names, posts, dates, locations, etc. Using various combinations of blocks and intermediate syntactic elements allows to build a great variety of patterns. Exact position of elements corresponding to the "exact answer" was localized within the structure of each pattern. Each pattern is characterized by a generalized semantics, thus the pattern-matching string must be checked for correlation with the question terms and/or their synonyms/substitutes.

Essentials of the Approach

In 2002 TREC QA track tests we have further developed the approach described in [Soubbotin, 2001]. In general, our method lies in the domain of approaches examining the potential of information extraction for question answering tasks [Srihari, Wei Li, 1999; De Boni, 2001]. The evolution of IE systems, as represented, in particular, at Message Understanding Conferences (MUCs), shows a certain shift from deep text analysis based on computational linguistic and NLP methods to surface techniques [Eagles, 1998]. Our approach can be considered as being in line with this tendency.

More specifically, our approach is based on the use of formulas describing the structure of strings likely bearing certain semantic information. For example, string "FBI Director Louis Freeh" can be recognized, according to one of such formulas, as likely bearing the following information: a person represented by his/her first and last names occupies a (leading) post in an organization. The formula for this string is: a word composed of capital letters; an item from the list of posts in an organization; an item from the list of first names; a capitalized word. We can mark two first items in this formula as "exact answer", if we want to get answer to the question "Who is Louis Freeh?", and two last items, if the question is "Who is FBI head?" (question 1583 at TREC 2002).

First used at TREC-10 QA track, formulas of such kind were called "patterns" [Soubbotin M.M and Soubbotin S.M, 2001]. The term "pattern" is widely used in the field of Information Extraction. Our concept of patterns as structural formulas for strings is obviously different from that in "traditional" IE field, but keeping this difference in mind, we consider it convenient to use this term.

Each pattern is characterized by a certain generalized semantics, because the formulas' items refer to certain semantic categories (e.g., "posts") and not to specific semantic units (e.g., "president", "head", "director"). Therefore, after a string corresponding to a formula is recognized, the next step is to identify the question terms (or their synonyms/substitutes) within it or in its surrounding. To increase the likelihood of getting the right answer, the surrounding of the found string must be checked for the presence of expressions negating its semantics (e.g., "former", "-elect", "deputy", etc., located before or after the term from the list of posts).

After a question's type is defined (e.g., question about a person occupying certain post in an organization, question about husband/wife/relative of a person, question about acronym, etc.), a set of formulas, prepared for this type, is applied to match the strings in question-relevant passages.

Our approach does not need to distinguish linguistic entities in the text. We handle the source text strictly as string, i.e. consisting only of characters. The patterns used in our QA approach are aimed only at recognizing sequences of elements that correspond to the predefined formulas.

As surface patterns, our formulas for strings are similar to wrappers [Adams, 2001; Kushmerick, 2000] and look like regular expressions. However, patterns used by the wrapper techniques are mostly resource-specific, they relate to the document formats rather than the ways information is presented in written texts per se. As for difference from regular expressions, it is worth noting that patterns, that we use, include elements referring to the lists of predefined words/phrases.

Currently, increased attention is seen on surface approaches in QA. In some recent publications surface patterns similar to those used by us were discussed [Magnini, et al., 2002; Brill, et al., 2002; Brill, et al., 2001; Ravichandran and Hovy, 2002; Hovy et al., 2002].

Patterns and Question Types

The IE task, as presented at its main forum - the Message Understanding Conferences (MUCs), is focused on certain topics, or domains (Terrorism, Management Successions, Natural Disasters, Outbreaks of Infectious Diseases, etc.). The QA task requires another way to categorize the addressed Information.

The usual praxis of TRECs' QA tracks participants is to predefine a set of potential question types. The questions accumulated from several TRECs represent a good source for defining question types on a more or less detailed basis. The paradigm of "information categories" defined by question types (in contrast to "topic/domain" paradigm) allows to create systematically a variety of patterns, basing on potential semantic relationships inside each question category.

So, for the question type "Who is person X?" we can presuppose - among the main alternative possibilities - that this person is known for the (top-level) position he/she occupies in a organization, company or government; for his/her contributions as author, inventor, founder, etc.; as outstanding figure in a professional area; as wife/husband/relative of a well-known person; as involved in well-known event (e.g., as a criminal/perpetrator). In each case, a relationship is established between two or more entities: person, post, and organization/company; author and work; etc. The same entities are present if the Who-questions refer to posts, authors, etc. (e.g., "Who occupies the post Y in the organization Z?".)

For most Where-questions, we can suggest geographical items as answers. This is achieved by constructing structural formulas like: item from the list of cities/towns/counties, etc.; comma; item from the list of countries/states. There are question types suggesting as answers combinations of digits with units of measurement or currencies names. Completeness of lists corresponding to "semantic" pattern elements is evidently important (e.g., the list of currencies must include not frequently used words, such as "dlrs").

The type of the processed question is defined basing both on its interrogative and on the presence of words/expressions that are included in the list of characteristic terms for the corresponding question type.

Complex Patterns

Complex patterns are formulas for strings expressing relationships between several semantic entities. There are some basic, typical, frequently used ways of expressing certain relationships between semantic entities in written texts of a language covering the most ways these relationships are expressed in text corpora. There are also less usual ways for expressing these relationships. (Our preliminary investigations show that information on companies' leaders (name, post, company) in more than a half of cases is expressed by strings corresponding to 5 main groups of our complex patterns). Thus, one can gradually embrace less frequent string structures ensuring the more complete covering of certain relationships by a set of patterns.

Each basic way to express a predefined relationship between semantic entities has a great variety of variants. For example, blocks corresponding to people names can include items from first names list, capitalized words, specific name elements such as "bin", "van", etc., capital letters, dots, abbreviations like "Sr." and "Jr.". Multiple ways of writing people's names are reflected in corresponding block formulas (e.g., only first names - for children and pets; combinations of first, middle and last names; people's names of various nationalities; names wits initials, etc.). Blocks corresponding to dates are composed from prepositions, articles, digits, month names, commas, dashes, brackets, special words/phrases like "early", "in the period of", "years ago", "B.C.", etc.

As some non-obligatory elements can be present or not in the corresponding strings, it is important to foresee the most complete set of possible variants for each basic pattern.

Semantic entities (e.g. personal and organizations' names, posts, locations) can be represented in a complex pattern by lists of elements with explicit semantics (words/phrases fitting in a corresponding semantic category), as well as by elements that do not per se bear any definite semantics (e.g., capitalized words); these elements can represent – with higher or lower probability - a certain semantic entity due to their presence in the complete pattern structure.

The validity of a pattern is dependent of its elements and structure.
According to our observations, the more complex a pattern's internal structure, the higher is its validity (reliability). As a rule, complex patterns containing many elements are more valid: the neighboring elements mutually confirm each other. If pattern elements corresponding to a person's name include only such indicators as capitalized words, validity of this pattern is low.

But if a capitalized word is preceded by an item from the list of first names plus an item from titles list, then the validity increases significantly. For resolving the ambiguity of capitalized words - whether they are proper names - the system recognizes the usually capitalized words, such as names of months, etc. (For issues of proper names identification see, for example, [Bikel, 1997; Viitanen, 2002]).

However, for some question types, structurally simple, small-sized patterns can be used to match answers for these questions with high likelihood (e.g., a sequence of four digits for "In what year"-questions; digits plus units of measurement for questions regarding length, area, weight, speed, etc.) So, for answering the question 1634 "What is the area of Venezuela?", a simple pattern allows to match the string "340,569 square miles".

Correlation between the Pattern-Matching String and the Question Semantics

As said above, multiple strings can correspond to each pattern structure. The suitable string can be recognized if words/phrases of the question (or their synonyms/substitutes) are present inside this string or In its surrounding. If a string is matched by complex, multi-elements pattern, the presence of question terms can be checked within it at certain predefined positions. This simplifies the task of verifying the suitability of a matched string. By contrast, strings matched by small-sized patterns usually do not contain all the terms expressing the semantics of a given question type. In this case, to verify sufficiency of correlation between a pattern-matching string and semantics of a question, the surrounding of the pattern must be explored.

The simpler a pattern's structure, the more significant is how question words are located in the surrounding of the pattern-matching string, i.e. at which distance, right or left to the pattern, in which position to other potentially present pattern-matching strings, etc. Other important factors are the number and total weight of question words present in the pattern's surrounding. The weight (rank) assigned to a question word/phrase (or to its substitute) was defined basing on its relative "specificity" in the documents corpus. The highest rank was assigned to quoted expressions and (chains of) capitalized words. Specificity of other words was determined basing on their occurrence frequency in the corpus.

Thus, the relative simplicity of a pattern's structure is compensated by the complexity of rules that should verify the candidate answer's correlation with the question semantics. For each question type, the patterns are grouped into two subsets: complex and (relatively) simple.

We think that the straightforward use of surface patterns for QA without applying a set of heuristic rules for checking the patterns surrounding (see [Hovy et al., 2002]) cannot ensure sufficiently reliable results.

The total score assigned to candidate answers is based both on pattern's reliability and on evaluation of question words's presence inside a pattern-matching string or in its surrounding.

Overview of the QA Process

The process flow includes the following main stages.

Defining the question types for all questions - basing on interrogatives and on the lists of characteristic terms for the corresponding question type.
Ordering the questions aimed at first processing the question types for which there are more reliable patterns.
Forming the query from question terms; ranking query terms according to their "specificity".
Modifying the query, if the search failed or if an answer's score is beneath a predefined threshold (single words can be used instead of phrases; terms from lists of substitutes are added).
Identifying the pattern-matching strings for this question type - applying first a set of complex patterns, then a set of simple patterns.
Checking for correlation between the pattern and the question's semantics.
Identifying the exact answer part in the pattern-matching string.
Calculating the total score for each candidate answer.
Selecting the top-ranking candidate.
Creating a record for the submission file.

Analysis of the Results

Analysis of our successes and failures at TREC 2002 allows to see some characteristic peculiarities of patterns approach for QA.

Our confidence-weighted score is 0.691. The way the obtained answers were ordered was based on the predefined order of question types. So, we have suggested that a simple but highly reliable pattern for questions of the types "(In) what year" and "When" will match in most cases the right strings (taken into account the correlation with the question semantics). As a result, our first 29 recognized answers belonged to questions of these types, among which only 3 answers were wrong. Of course, this influenced positively our confidence-weighted score.

Noteworthy, our answer to the question 1617 ("When did the Klondike gold rush occur?") was assessed as wrong. Our answer "1896" was based on the presence of this string in the sentence containing 3 from 4 question words: "In 1896, a prospecting party discovered gold in Alaska, a finding that would touch off the Klondike gold rush." Another this group answer assessed as wrong is "Victorian era" to the question "When was Benjamin Disraeli prime minister?" (the answer was got from the sentence "Benjamin Disraeli was the most famous Conservative leader of the Victorian era"). These examples show that answers obtained by use of patterns, even if they are not correct, are not senseless, and in many cases are semantically close to right answers. We consider this feature as important for real use of a patterns-based QA system.

Some answers assessed as unsupported demonstrate the same feature of the pattern method. To the question 1476 ("Who was the Roman god of the sea?") correct answer "Neptune" was obtained by matching the string "Neptune, the god of the sea". This string was present within the sentence describing the decoration of a building and was assessed as unsupported apparently on this ground. We think that the possibility to extract the right answer from non-relevant documents/passages, in fact, can be regarded as extending the capacity of the QA system.

12 answers were assessed as "inexact." The exactness of answers (as well as the percentage of right answers) can be increased by further completing the library of patterns and lists of predefined words/phrases. For question about the cost of the international space station we

obtained the answer "dlrs 40 billion"; the exact answer is "at least dlrs 40 billion." Our patterns for this question type include such blocks as currencies names, digits, numerals, and items from the list of adjusting expressions ("more than", "not less than", etc.); the expression "at least" was missing in this list.

The number of right answers was 271. From the 209 wrong answers 148 were "no answer". In the vast majority of these cases the passages where the answer strings might be matched were not found. This was mainly due to that the system was working primarily with the top 50 documents collections supplied for each question. Excluding the "NIL" answers, we can evaluate the rate of wrongly identified answer strings: 48 for 352 questions (13,6%).

After the end of the TREC test we have upgraded our QA system to process large documents collections more efficiently. Now, selective processing of questions to which the answers had not been recognized shows that, to a great extent, the right answers can be obtained instead of wrong "NIL."

Further Work

The similarity between the TREC-11 QA task (that was focused on getting exact answers) and information extraction tasks was an incentive for use of our surface patterns in the framework of the IE technology. Using a modification of the approach applied at TREC-11 QA tests we developed a domain-independent system that extracts information from unstructured texts and populates a database. This system, named "ExactAnswer", identifies entities such as persons, organizations, locations, and other types of data as well as relationships between entities (e.g. persons in relation to organizations). The tests conducted on various kinds of unstructured texts show high degree of accuracy (over 95%).

Adding more power to the patterns method remains our continued task. We use patterns also in the software products that are developed in the framework of our long-term project (http://insight.com.ru/), aimed not only at extracting of text units, but also at combining them into complex structures, such as single-document and multi-document summaries, discourse and reasoning chains. We also intend to examine the theoretical aspects of patterns considered as structural formulas for text strings. Primarily, we mean a specific dimension of studying languages - as they are represented in written texts - aiming at revealing correlations between the structure of text strings and their semantics.

References

Adams, Katherine C.. The Web as a Database. New Extraction Technologies & Content Management. Online, March/April 2001, pp. 28 - 32.

Bikel, D.M., et al., 1997. Nymble: a High-Performance Learning Name-finder. Proceedings of the Fifth Conference on Applied Natural Language Processing,Morgan Kaufmann Publishers, pp. 194-201.

Brill, Eric, Jimmy Lin, Michele Banko, Susan Dumais and Andrew Ng. Data-Intensive Question

Answering.
http://www.ai.mit.edu/people/jimmylin/publications/Brill-etal-TREC2001.pdf

Brill, Eric, Susan Dumais and Michele Banko. An Analysis of the AskMSR Question-Answering System.
http://research.microsoft.com/~sdumais/EMNLP_Final.pdf

De Boni, Marco. Information Extraction, Query-Relevant Summarization and Question Answering: an Overview. 2000-2001.

EAGLES. Preliminary Recommendations on Semantic Encoding. Interim Report. Information Extraction, May 1998.
http://www.ilc.pi.cnr.it/EAGLES96/rep2/node30.html

Hovy, Eduard, Ulf Hermjakob, Deepak Ravichandran. A question answer typology with surface text patterns.
http://www.isi.edu/~ravichan/papers/hlt2002isi.pdf

Kushmerick, Nicolas. Wrapping up the Web.
Synergy: Newsletter of the EC Computational Intelligence and Learning Cluster Issue 2 (Spring 2000) http://www.dcs.napier.ac.uk/coil/news/feature46.html

Magnini, Bernardo, Matteo Negri, Roberto Prevete, and Hristo Tanev. Towards Automatic Evaluation of Question/Answering Systems.
http://tcc.itc.it/research/textec/topics/question-answering/lrec2002.pdf

Ravichandran, Deepak and Eduard Hovy. Learning Surface Text Patterns for a Question Answering System. Proceedings of the ACL Conference, 2002.

Soubbotin, M. M. and Soubbotin, S.M. Patterns of Potential Answer Expressions as Clues to the Right Answers. TREC Proc., 2001.

Srihari, Rohini K. and Wei Li. Information Extraction Supported Question Answering. TREC, 1999.

Viitanen Sirke. Named entities in BRIEFS. 2002.
http://www.ling.helsinki.fi/users/stviitan/prosem.html

Novelty track at IRIT – SIG

Taoufiq Dkaki[1,2], Josiane Mothe[1,3], Jérôme Augé [1]

(1) Institut de Recherche en Informatique de Toulouse, 118 Rte de Narbonne, 31062 Toulouse CEDEX

(2) IUT URS, 72 Rte du Rhin, 67400 Strasbourg

(3) Institut Universitaire de Formation des Maîtres Midi-Pyrénées, 56 av de l'URSS, 31078 Toulouse CEDEX

Abstract

IRIT developed a new strategy in order to detect the *relevant sentences* that we did not try in a more general context of document retrieval but did try previously and partially in document categorization. In our approach a sentence is considered as relevant if it matches the topic with a certain level of coverage. This level of coverage depends on the category of the terms used in the texts. Three types of terms have been defined: highly relevant, lowly relevant and no relevant. With regard to the *novelty part*, a sentence is considered as novel when its levels of coverage with the previously processed sentences and with the best-matching sentences do not exceed certain thresholds.

1 Introduction

«The TREC 2002 novelty track is designed to investigate systems' abilities to locate relevant and new information within the ranked set of documents retrieved in answer to a TREC topic » [trec.nist.gov].

Retrieving relevant texts is traditionally based on computing a similarity between the representation of the information need (or topic) and the texts. This general statement has been applied to full documents as well as chunks of texts (passage retrieval). Intuitively, the same idea can be applied when sentences retrieval is involved. IRIT developed a new strategy in order to detect the relevant sentence that we did not try in a more general context of document retrieval but did try previously and partially in document categorization. In our approach a sentence is considered as relevant if it matches the topic with a certain level of coverage. This level of coverage depends on the category of the terms used in the texts. Three types of terms have been defined: highly relevant, lowly relevant and no relevant. With regard to the novelty part, a sentence is considered as novel when its levels of coverage with the previously processed sentences and with the best-matching sentences do not exceed certain thresholds.

The results we obtain are quite good for the 'relevant' part. Indeed, we obtain 36 topics (73%) for which the R*P is higher or equal to the average of the 42 runs. With regard to the 'novelty' part, the results are disappointing and our method is situated around the average.

This paper is organized as follows: in section 2 we describe the method we used, including the way documents and topics are represented and the strategies we developed for the two sub-tasks (relevant part and novelty part). In section 3 we present the results and comment them. Finally we indicate in the last section the future directions for our work on novelty track.

2 Description of the method

2.1 Document and topic representation

In our method, topics and sentences are considered as texts. Each text is pre-processed the same way in order to extract the representative terms. Then, the terms extracted from the topics are categorized into two groups : highly relevant terms (HT) and lowly relevant terms (LT). Finally, each text is represented by these two set of terms, with weights associated to each term.

2.1.1 Text processing

Texts are processed using the following method :

1. Stop words are removed,
2. The remaining words are normalized using a dictionary that provides a common root for different words. This dictionary contains 21291 entries.

2.1.2 Topic processing

A topic is considered as a single text and the representative terms are extracted as explained in the previous section. Each term is then weighted and categorized into 2 groups:

- Highly relevant terms are terms that get a weight greater than 3,
- Lowly relevant terms are terms that get a weight equal to 1 (see below the formula used to compute the term weights).

Given T_k a topic, t_i a term and $tf_{i,k}$ the frequency of t_i in T_k.

The term weight regarding a topic is computed as follows:

$$\begin{aligned} weight(t_i,T_k) &= tf_{i,k} & \quad if \quad tf_{i,k} \geq 3 \\ &= 1 & \quad otherwise \end{aligned}$$

In order to obtain a significant difference -in terms of importance- between the highly relevant terms and the lowly relevant terms the weights of the lowly relevant terms are set to 1.

Each term is also categorized into one of the groups defined as follows:

$$HT_k = \{t_i \,/\, t_i \in T_k \text{ and } weight(t_i,T_k) > 1\}$$

$$LT_k = \{t_i \,/\, t_i \in T_k \text{ and } weight(t_i,T_k) = 1\}$$

2.1.3 Document processing

Each sentence of a document is considered as a text and the representative terms are extracted as explained in the section 2.1.1. To each term is associated a weight defined as follows:

Given S_j a sentence, t_i a term and $tf_{i,j}$ is the frequency of t_i in S_j.

$$weight(t_i,S_j) = tf_{i,j}$$

2.2 Relevant sentences

In order to decide if a sentence is relevant, we associate three components to each sentence :

- a score that reflect the sentence – topic matching :

Given a topic T_k and a sentence S_j

$$\begin{aligned} Score(S_j,T_k) &= \sum \left(weight(t_i,S_j) \cdot weight(t_i,T_k)\right) \\ &= \sum_{t_i \,/\, t_i \in HT_k} \left(tf_{i,j} \cdot tf_{i,k}\right) + \sum_{t_i \,/\, t_i \in LT_k} \left(tf_{i,j}\right) \end{aligned}$$

- and two groups of terms:

$$HS_j = \{t_i \,/\, t_i \in (Sj \cap HT_k)\}$$
$$LS_j = \{t_i \,/\, t_i \in (Sj \cap LT_k)\}$$

HS_j corresponds to the highly relevant terms from the topic that occurs in the sentence,

LS_j corresponds to the lowly relevant terms from the topic that occurs in the sentence,

A given sentence S_j is then considered as relevant iff :

$$Score\,(S_j, T_k) > f\left(\frac{|LS_j|}{|LS_j| + |HS_j|}\right) \cdot |HT_k| + g\left(\frac{|HS_j|}{|LS_j| + |HS_j|}\right) \cdot |LT_k|$$

where $|X|$ is the number of elements of X

In the experiments that correspond to the run sent to TREC, the function $f(\,)$ and $g(\,)$ have been set to:

$$f(0) = 2\,;\ \forall x \in \left]0.1\right],\ f(x) = 1.5$$
$$g(0) = 0.85\,;\ \forall x \in \left]0,1\right],\ g(x) = 0.3$$

2.3 Novelty sentences

To decide if a sentence p is to be considered as novel, we compute the similarity between the sentence p and the previous processed sentences p_i and the similarity between the sentence p and a sentence P' automatically built from the union of the set of p_i:

Given

$P = \{p_1, p_2, \ldots, p_n\}$ a set of sentences and $P' = \bigcup_{i \in \{1,\ldots,n\}} p_i$, P' is a sentence made of the set of sentences P,

$Sim(x, y\,)$ a function that compute a similarity between x and y and

p a sentence for which the system has to decide if it brings something new.

We first compute the following similarities:

$Sim(p, P') = \alpha_p$ and for $i \in \{1, \ldots, n\}$ $Sim(p, p_i) = \omega_{p,i}$

We then consider the q best previous sentences:

$for\ i \in \{1, \ldots, n\}$ $\delta_{p,i}$ is the series obtained by ordering $\omega_{p,i}$ in decreasing order.

$\beta_p = \sum_{i \in \{1,\ldots,q\}} Sim(p, \delta_{p,i})$ where q=4 in the run sent to TREC.

p is considered as novel iff:

$$\alpha_p \geq T_1 \text{ and } \beta_p \geq T_2$$

where $T_1 = 1$ and $T_2 = 0.6$ for the run sent to TREC.

3 Results

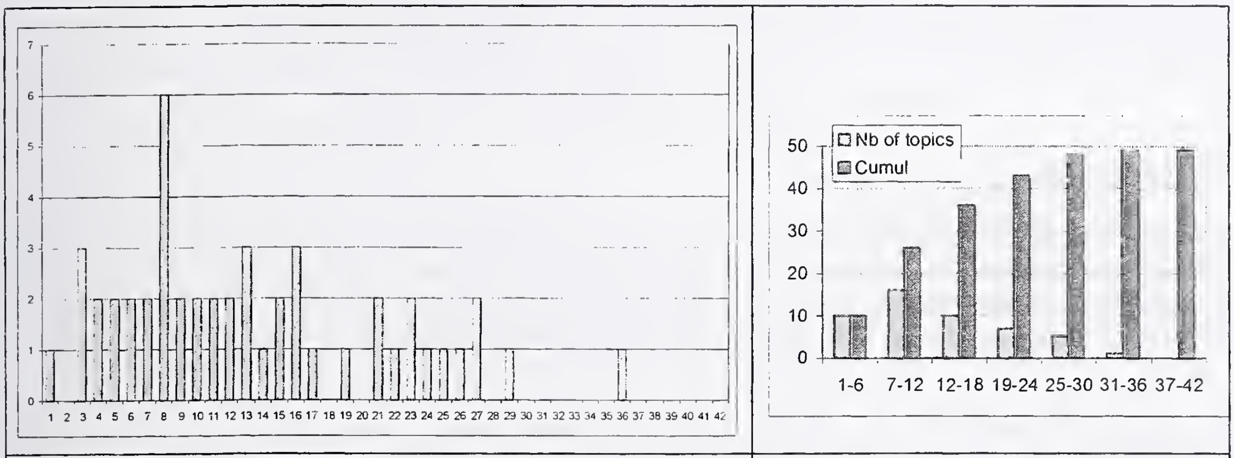
This section presents the results we obtained with the method we developed and using the parameters that have been described in section 2.

When comparing the results with the other runs, we can notice that our system is better in finding relevant sentences than in detecting novelty in the sentences. This can be explained by the method we used that does not take into account the order of the sentences in the documents. Additionally most of the parameters have to be tune specifically to take into account the sentence relationship.

3.1 Relevant sentences

Figure 1 indicates the number of topics for which our system (or run) has been ranked at the Xth position among the 42 runs. For example, our method obtains the best results for 1 topic, the third position for 3 topics, the fourth for 2 topics, etc. and has a rank higher than 30th for only one topic (see figure 1.a). Figure 1.b provides a graph that summarize figure 1.a by grouping together the results obtained for ranges of ranks. Additionally, the cumulative number of topics per range of system position is provided on the same graph. For example, we obtained a rank between 1 to 6 for 10 topics. The system obtains a rank equal or higher than 24 for 43 topics.

This clearly shows that our method is better than the average of the results. To be more precise, over the 49 topics, we obtained 36 topics (73%) for which the R*P is higher or equal to the average of the 42 runs. And if we consider the run ranks, we obtained a rank higher than the middle (21) for between 37 and 39 topics (depending how we consider the rank when 2 systems obtained the same value for R*P).



a) Number of topics per run rank : detailed results

b) Number of topics per run rank : summarized results

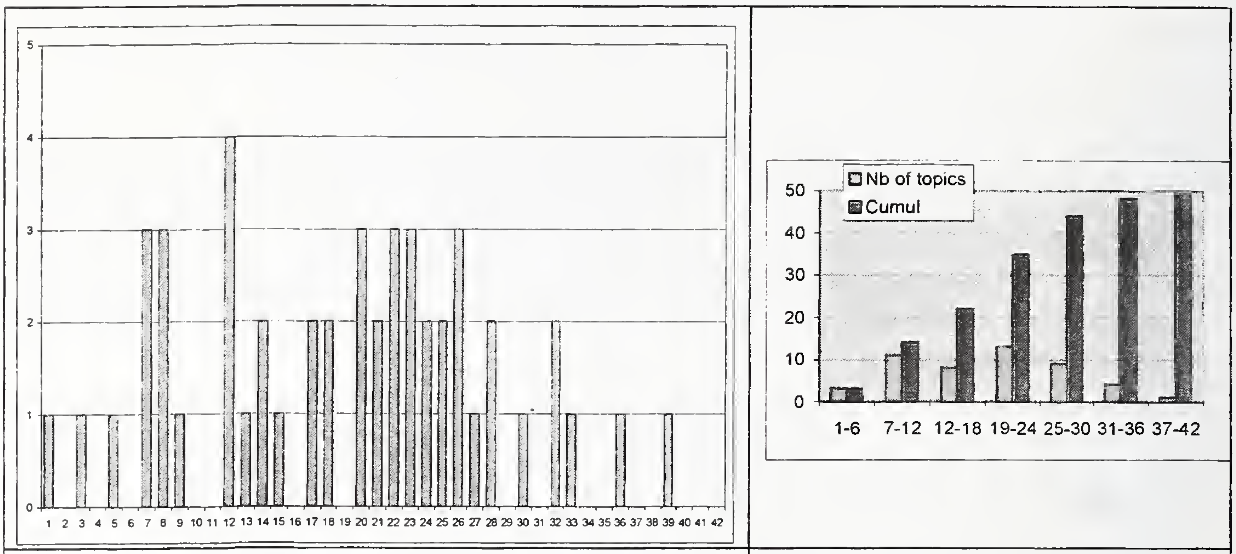
Figure 1 : Number of topics per run rank – relevant sentences

3.2 New sentences

We present the results obtained in the second subtask the same way (see Figure 2).

Over the 49 topics, we obtained 20 topics (about 40%) for which the R*P is higher or equal to the average of the 42 runs. And if we consider the run ranks, we obtained a rank higher than the middle (21) for between 23 and 27 topics (depending how we consider the rank when 2 systems obtained the same value for R*P). The method is not better than the average.

The results in the second subtask are directly linked to the results obtained in the first subtask. As a result, the groups that obtained high R*P in the first part are more likely to obtain good results in the 'novelty' part. We can consider that the results we obtained for the novelty part are quite disappointing as the results on the relevance part were good. One of the tracks_we are going to explore to improve the results is to take into account the order of the sentences as two sentences from a same paragraph are more likely to treat the same subject for example.



a) Number of topics per run rank : detailed results

b) Number of topics per run rank : summarized results

Figure 2 : Number of topics per run rank - novelty

4 Conclusion

The approach we developed leads to relevant results for the first part of the task (relevant sentences). Over the 49 topics, we obtained 36 topics (73%) for which the R*P is higher or equal to the average of the 42 runs. And if we consider the run ranks, we obtained a rank higher than the middle (21) for between 37 and 39 topics. With regard to this sub-task, future work will be devoted first improving the definition of the $f(\,)$ and $g(\,)$ functions, which play the role of thresholds.

With regard to the second sub-task (novelty), the results are just on the average. This can be explained by the method we used that does not take into account the order of the sentences in the documents. Additionally most of the parameters have to be tune specifically to take into account the sentence relationship (two sentences that are close together in a document are more likely to deal with the same subject). This probably will also improve the fist sub-task of novelty track.

5 References

[trec.nist.gov] TREC web site.

[Luhn, 60] Luhn, H., Keyword in Context Index for Technical Literature, American Documentation XI (4), 1960, 288-295.

[Mothe, 02] Mothe J., Chrisment C., Dousset B., Alaux J., DocCube : multi-dimensional visualisation and exploration of large document sets, to appear in JASIST.

IRIT at TREC'2002: Filtering Track

M. Boughanem, H. Tebri and M. Tmar

IRIT/SIG
Campus Univ. Toulouse III
118, Route de Narbonne
F-31062 Toulouse Cedex 4
Tel: 33 (0) 5 61 55 74 16

Email: $\{boughane, tebri, tmar\}@irit.fr$

Abstract

The experiments we undertaken this year for TREC2002 Filtering track, are focussed on threshold calibration. We proposed a new approach to calibrate the dissemination threshold in an adaptive information filtering. It consists of optimizing a utility function represented by a linearized form of the probability distributions of the scores of the relevant and the non-relevant documents already filtered. The profiles are learned using the same method used last year. It is based on a reinforcement algorithm. We submitted results on three tasks: adaptive, batch and routing.

1 Information representation and filtering

Our adaptive filtering model is inspired from the connectionist model Mercure [1]. The profile and the document are represented by a set of weighted terms. The filtering process consists of computing a relevance value *RSV Retrieval Status Value*. The document is delivered only if the *RSV* is greater than the dissemination threshold. A learning process is then carried out by modifying the profile and the dissemination threshold to be more efficient in the future.

1.1 System initialization

The user profile is represented by a set of terms :

$$p^{(0)} = \left(\left(tp_1, w_1^{(0)}\right), \left(tp_2, w_2^{(0)}\right) \ldots \left(tp_n, w_n^{(0)}\right)\right) \tag{1}$$

where tp_i is a term and $w_i^{(0)}$ is its weight in the initial profile (at $t = 0$). in the rest of this paper, the term-profile weight is noted $w_i^{(t)}$ where t represents the instant when the system receives a document. Initially, the term-profile weight is computed as follows:

$$w_i^{(0)} = \frac{tfp_i}{max_j\,(tfp_j)} \tag{2}$$

Where tfp_i is the term frequency of tp_i in the profile. The formula seems to be abusively simplistic, but at the beginning of the filtering process, no information is known but a set of terms and their occurrence frequencies in the initial user profile. However, this weight will be adjusted by learning.

1.2 Filtering incoming documents

Each incoming document at time t is indexed, to build a list of stemmed terms (Porter [3]). the terms belonging to a stop-list are removed. Each term is then weighted according to the following formula:

$$d_i^{(t)} = \frac{tf_i^{(t)}}{h_3 + h_4 * \frac{dl^{(t)}}{\Delta l^{(t)}} + tf_i^{(t)}} * log\left(\frac{N^{(t)}}{n_i^{(t)}} + 1\right) \tag{3}$$

Where $tf_i^{(t)}$: term frequency t_i in the document $d^{(t)}$, h_3, h_4: constant parameters, for the experiments $h_3 = 0.2$ and $h_4 = 0.7$, $dl^{(t)}$: document length $d^{(t)}$(number of index terms). $\Delta l^{(t)}$: average document length,

$N^{(t)}$:number of incoming documents until time t, $n_i^{(t)}$: number of incoming documents containing the term t_i. This weighting formula is a form of *Okapi* [5] term-query weight used in the information retrieval system *Mercure*.

$N^{(t)}$, $n_i^{(t)}$ and $\Delta l^{(t)}$ are collection based parameters that are computed on the cumulative documents filtered at time t. They can not be known while the document stream does not stop sending documents, the used values are recomputed for each incoming document.

A relevance value noted $rsv\left(d^{(t)}, p^{(t)}\right)$ is then computed corresponding to the document and the profile, as follows:

$$rsv\left(d^{(t)}, p^{(t)}\right) = \sum_{t_i \in d^{(t)}, tp_j \in p^{(t)} \ and \ t_i = tp_j} d_i^{(t)} * w_j^{(t)} \tag{4}$$

The binary decision rule used for document filtering is the following:

$$\begin{cases} if \ rsv\left(d^{(t)}, p^{(t)}\right) \geq threshold^{(t)} \ accept \ the \ document \\ otherwise \ reject \ the \ document \end{cases}$$

The threshold value is initially fixed to a low value, 0 in our experiment. It is modified while filtering.

1.3 Profile adaptive learning

The learning process adopted in our case is incremental. It is processed at each filtered document judged as relevant by the user. The basic idea of our learning process is based on the relevance reinforcement process. When a document is judged as relevant, it is necessary to be able to find a new representation of the profile which makes it possible to find the document with a strong score. In other words, one will be brought to improve the profile such as $rsv\left(d^{(t)}, p^{(t)}\right) = \beta$ where β is the desired score rsv. This technique was used in TREC-10, we do not give more details, the reader can be refer to TREC-10 [1].

The problem is assimilated to a linear equation resolution. Each solution of this system is a set of weights affected to the document terms, knowing that there is an infinite number of solutions, we add a constraint to obtain only one solution. The system to resolve is given by :

$$\begin{cases} \sum_{\substack{t_i \in d^{(t)} \\ tp_j \in p^{(t)} \\ t_i = tp_j}} d_i^{(t)} w_j^{(t)} = \beta \\ \dfrac{w_i^{(t)}}{f\left(d_i^{(t)}, r_i^{(t)}, s_i^{(t)}\right)} = \dfrac{w_j^{(t)}}{f\left(d_j^{(t)}, r_j^{(t)}, s_j^{(t)}\right)} \forall \left(t_i, t_j\right) \in {d^{(t)}}^2 \end{cases} \tag{5}$$

The solution of the system 5 is a set of "provisional" weights, let n be the number of different terms in the processed document at time t, $f_i^{(t)} = f\left(d_i^{(t)}, r_i^{(t)}, s_i^{(t)}, R^{(t)}, S^{(t)}\right) \forall i \in \{1 \dots n\}$ where $r_i^{(t)}$(resp. $s_i^{(t)}$) is the number of relevant (resp. non-relevant) documents containing the term t_i until the time t. For each term appearing in the document, the provisional weight solution of the system 5 is the following:

$$\forall i, \ pw_i^{(t)} = \frac{\beta f\left(d_i^{(t)}, r_i^{(t)}, s_i^{(t)}\right)}{\sum_j f\left(d_j^{(t)}, r_j^{(t)}, s_j^{(t)}\right) * d_j^{(t)}} \tag{6}$$

The function f is proportional to the term importance. The function f used for the experiments is the following:

$$\forall i, f\left(d_i^{(t)}, r_i^{(t)}, s_i^{(t)}\right) = d_i * log \frac{\frac{r_i^{(t)} + 0.5}{R^{(t)} - r_i^{(t)} + 0.5}}{\frac{s_i^{(t)} + 0.5}{S^{(t)} - s_i^{(t)} + 0.5}} \tag{7}$$

$R^{(t)}$ (resp. $S^{(t)}$) is the total number of relevant (resp. non-relevant) documents until the time t.

The "provisional" weight $pw_i^{(t)}$ contributes in learning the term-profile weight corresponding to the term t_i, we use the following gradient propagation formula:

$$w_i^{(t+1)} = w_i^{(t)} + log\left(1 + pw_i^{(t)}\right) \tag{8}$$

We add 1 to $pw_i^{(t)}$ to avoid adding negative value to the term-profile weight.

1.4 Threshold calibration

To find the best threshold, the system should follow the document score evolution and regulate the threshold in order to select as maximum as possible the relevant documents and reject as maximum as possible the non-relevant documents. The threshold regulation can be made by maximizing a utility function[1]. According to the sampling theory, the behavior of a random sample is the same of all the population, so the threshold allowing to maximize the utility function in a random sample of documents allows to maximize the same utility function in all the documents of the stream. The approach we propose for threshold calibration consists of estimating the discrete probabilities of the scores of the relevant and the non-relevant documents already filtered and then using a kind of probability plotting method to build a linearized probability density distribution. The threshold that maximizes a utility function, represented by both distributions, is then selected.

1.4.1 Score distribution modelling

The probability that a random document has a particular score is equal to the number of documents having the same score divided by the total number of documents:

$$p(X = score) = \frac{|\{d|rsv(d,p) = score\}|}{|\{d\}|} \tag{9}$$

As score values are very distinct, they tend to be equiprobable ($|\{d|rsv(d,p) = score\}| = 1$ or 0). Indeed, it is very difficult to find two or many documents having exactly the same score. Consequently the score probability distribution tends to be uniform.
Instead of computing the probability that a document has a score, we compute the probability that the document score belongs to an interval [2]. We define a set of intervals enough reduced so that the document scores belonging to the same interval are really close. We define n adjacent intervals $I_1, I_2 \ldots I_n$ having the same ray:

$$I_i \quad = \quad [score_{i-1}, score_i] \tag{10}$$

Where : $score_0 = \min_d rsv(d,p)$ and $score_n = \max_d rsv(d,p)$.
The number of intervals is proportional to the sample size, indeed the great is the number of documents, the great is the definition field of the document scores. We define n as the half of the total number of documents: $n = \frac{|\{d\}|}{2}$.
The probability that a document score belongs to an interval is given by:

$$p(score_i < X < score_{i+1}) = \frac{|\{d|score(d,p) \in]score_i, score_{i+1}[\}|}{|\{d\}|} \tag{11}$$

The representation of the probability distribution based on formula 11 corresponding to documents scores is poissonnian. There is many methods to estimate the probability law parameters followed by the document scores: the parametric regression and the maximum likelihood estimation method [4]. In both cases, the law must be assumed to be known in advance, these methods allow to estimate the parameters of this function. But, in an experimental context many limits of these methods can be noted. In deed, even though the assumption that the scores follow a known distribution density could be acceptable, but it strongly depends on the experimental conditions such as the filtering and the learning approaches and the size (number of documents) of the sample used for deriving the parameters of the distributions, the sample size must be enough important to obtain non-skewed estimations. To resolve these problems we propose that instead of assuming that the distributions are known, they are built by estimating the discrete probabilities of the scores of the relevant and the non-relevant documents delivered at a certain time and then by linearizing these probabilities to obtain the corresponding probability density distributions. A utility function, to be optimized, is then represented using these distributions. The best threshold is the threshold that maximizes this utility.

[1] *a function used to evaluate the filtering systems performance*

[2] *There is a method to estimate the parameters of a probability law allowing to define a more or less precise interval containing this parameter called confidence interval rather than a value which will be less probably equal to this parameter*

1.4.2 Probability distribution linearization

Based on the regression theory, the representative curve of any function can be linearized in order to facilitate many tasks (computing surfaces, searching an extremum ...). The linearization consists of considering the domain of the function and divide it into a set of intervals such as the representative curve of the restriction of that function in each interval can be assimilated to a linear curve.

We use this technique to linearize the representative curve of the probability density distribution of the scores. We assume that this function exists and we try to linearize it. As we do not know this function, we propose to linearize it using the corresponding discrete probabilities. The first stage of this process is to identify a set of linear intervals. We define a linear interval by two scores $[score_x..score_y]$ where $score_x < score_y$ and all the points formed by (s_i, p_i) fit a straight line, s_i is a score in that interval and p_i is the discrete probability of s_i computed according to Formula 11. The linearity of a set of points is measured using the least squares method [4]. The least squares method requires that a straight line be fitted to a set of points such that the sum of the squares of the distance of the points to the fitted line is minimized. In our work the detection of linear interval is measured incrementally by considering all the points (s_i, p_i), ordered in increasing order of the scores. It consists of adding a point to a given set of points representing a straight line and computing an error which measures the standard deviation between that "new" set and a linear curve. If this error is below than a "linearity threshold", the considered point is definitely added to that set, the next point is then considered. Otherwise this point is removed from the set, and we continue the search of a new linear interval, and so on until the last point of the distribution. The following algorithm is applied:

1. $c = 1$ *(c is the index of the classes of the points with scores within a linear interval)*
2. $P = \emptyset$,
3. $threshold\_error = 0.0001$.
4. $< M + 1$ *: the total number of the points of the form* (s_i, p_i)*. /*We consider that the points are ranked in increasing order of their scores.* $i = 0$ *in the first score.*
5. *for* $i \in \{0 \ldots M\}$.
   (a) $P \leftarrow P \cup \{i\}$.
   (b) *determine the equation of the line* $D_c : y(x) = a + bx$ *based on the linear regression for all points* $(s_j, p_j) \forall j \in P$,
   (c) *compute the standard deviation error between the points* $(s_j, p_j) \forall j \in P$ *and the line* D_c:

$$E = \sum_{j \in P} d^2((s_j, p_j), D_c) \qquad (12)$$

$$d^2((s_j, p_j), D_c) = (\frac{a + b.s_j - p_j}{\sqrt{a^2 + 1}})^2 \qquad (13)$$

   (d) *if* $E > threshold\_error$.
      i. */*a class of points is formed.*
      $C_c = (d_c, f_c, a_c, b_c)$ *where.* $d_c = \min(s_j)$, $f_c = \max(s_j)$ $\forall j \in P$, a_c *and* b_c *are the coefficients of the equation of the line* $y = a_c + b_c x$ *derived using the linear regression of all the points* (s_j, p_j) *where* $j \in P | \{i\}$.
      ii. $P \leftarrow \{i\}$. */* re-initialize* P
      iii. $c \leftarrow c + 1$.
   (e) *end if*
   (f) *end for*

A transformation is necessary at this stage such that all lines form a continuous representation:

1. *Transform the representation into a continuous one by relying the extremities of two adjacent classes. This liaison is done as follows: for two adjacent linear classes C_c and C_{c+1}, rely f_c and d_{c+1} with a line having as equation $y = \alpha_c + \beta_c x$. This line should pass through the points $(f_c, a_c + b_c f_c)$ and $(d_{c+1}, a_{c+1} + b_{c+1}.d_{c+1})$, so:*

$$\alpha_c = \frac{a_c+b_c.f_c-a_{c+1}+b_{c+1}.d_{c+1}}{f_c-d_{c+1}} \tag{14}$$

$$\beta_c = a_c + b_c.f_c - \frac{a_c+b_c.f_c-a_{c+1}+b_{c+1}.d_{c+1}}{f_c-d_{c+1}} f_c \tag{15}$$

2. *Normalize the coefficients a_c, b_c, α_c and β_c such that:*

$$\int_{score_0}^{score_M} f(x)dx = 1 \tag{16}$$

The equation 16 is the fundamental property of the probability density functions. As $\int_{score_0}^{score_M} f(x)dx$ represents the surface formed by the graphical representation of f and X-coordinate axis, the coefficients a_c, b_c, α_c and β_c are divided by this surface. This surface becomes unit. This surface is computed as the sum of the surfaces formed by all the linear intervals and the X-coordinate axis, let cl be the number of linear classes, so:

$$\int_{score_0}^{score_M} f(x)dx = \frac{1}{2}(\sum_{c=1}^{cl}(f_c - d_c)(2a_c + b_c.(f_c + d_c)) + \sum_{c=1}^{cl-1}(f_c - d_{c+1})(2\alpha_c + \beta_c(f_c + d_{c+1})))$$

1.4.3 Threshold optimization

The proposed method is based on utility maximization. Generally the utility function is done by :

$$F = aR_+ - bS_+ \tag{17}$$

Where R_+ (resp. S_+) is the number of relevant (resp. non-relevant) selected documents.
Our goal is to determine a threshold allowing to maximize the theoretical value of F :

$$threshold^* = arg \max_{threshold} F \tag{18}$$

Where a, b : positive constants, R_+ : number of relevant selected documents, S_+ : number of non-relevant selected documents. R_+ and S_+ are both inversely proportional according to the threshold.

$$R_+ = p(r|score > threshold) * R \tag{19}$$

$$S_+ = p(s|score > threshold) * S \tag{20}$$

R and S represent the total number of relevant and non-relevant documents examined.
Based on Bayes transformation rule, we obtain :

$$R_+ = \frac{p(score > threshold|r) * p(score > threshold}{p(r)} * R \tag{21}$$

$$S_+ = \frac{p(score > threshold|s) * p(score > threshold}{p(s)} * S \tag{22}$$

$p(score > threshold|r)$ (resp. $p(score > threshold|s)$) represents the probability that a document is selected when it is relevant (resp. non-relevant). It represents the surface formed by the curve of function f corresponding to the relevant (resp. non-relevant) documents and the X-coordinate axis.

However, $p(r) = \frac{R}{N}$ (resp. $p(s) = \frac{S}{N}$) is the probability that a document is relevant (resp. non-relevant). Utility done by equation 18 is equivalent to:

$$\begin{aligned} F &= p(score > threshold) * N * (a * p(score > threshold|r) \\ &\quad + b * p(score > threshold|s)) \end{aligned} \tag{23}$$

The retained threshold value allows to maximize F.

2 Experiments and results

2.1 Results of adaptive filtering task

The pre-test documents were used to learn the profile and to set the initial threshold. Then the filtering process is processed on the test data. At each selected the relevant document, the profile is learned and the new threshold is also computed. The algorithm that has been used for this experiment is the following:

- if a document d^t is selected and is judged as relevant,
- learn the profile,
- re-estimate the scores of all delivered relevant documents and of a sample of 1000 non relevant documents extracted from the training data that have been used in batch,
- build the linearized probability distributions of these samples,
- Compute the new threshold that maximizes the utility function $T11U = 2R_+ - S_+$. This threshold is measured by varying the threshold value from the smallest score of the relevant documents to the greatest score of the non-relevant documents.

Table 1 lists the comparative adaptive filtering results for all topics.

<table>
<tr><th colspan="5">TREC adaptive filtering for topics 101-150</th></tr>
<tr><th>Evaluation</th><th>$= max$</th><th>$\geq median$</th><th>$< median$</th><th>Avg</th></tr>
<tr><td>$T11U$</td><td>2</td><td>31</td><td>19</td><td>0.386</td></tr>
<tr><td>$T11F$</td><td>1</td><td>28</td><td>22</td><td>0.387</td></tr>
<tr><td>Set precision</td><td>0</td><td>24</td><td>26</td><td>0.261</td></tr>
<tr><td>Set recall</td><td>0</td><td>30</td><td>20</td><td>0.409</td></tr>
<tr><th colspan="5">TREC adaptive filtering for topics 151-200</th></tr>
<tr><td>$T11U$</td><td>1</td><td>43</td><td>7</td><td>0.282</td></tr>
<tr><td>$T11F$</td><td>3</td><td>44</td><td>6</td><td>0.054</td></tr>
<tr><td>Set precision</td><td>3</td><td>44</td><td>6</td><td>0.092</td></tr>
<tr><td>Set recall</td><td>0</td><td>41</td><td>9</td><td>0.031</td></tr>
</table>

Table 1: Comparative adaptive filtering results

2.2 Batch and Routing Experiments

In batch and routing tasks the profile and the threshold were learned from the training collection. The learned profile and threshold were applied to the test data.

2.2.1 Batch filtering

We built a sample of 1082 documents, which corresponds to the documents of all the profiles which have been labelled as relevant in the training dataset. This sample is then used to learn the profile and the dissemination threshold. We only consider the 40 top terms. The learned profiles and thresholds were applied to the test database.
Table 2 lists the comparative batch results for all topics.

TREC batch filtering of topics 101-150				
Evaluation	= *max*	≥ *median*	< *median*	*Avg*
T11U	15	47	3	0.485
T11F	10	46	4	0.454
Set precision	9	47	3	0.661
Set recall	0	33	17	0.321

TREC batch filtering of topics 151-200				
Evaluation	= *max*	≥ *median*	< *median*	*Avg*
T11U	8	31	19	0.236
T11F	12	46	4	0.090
Set precision	21	47	3	0.288
Set recall	0	36	14	0.044

Table 2: Comparative batch filtering results

2.2.2 Routing track

We experiment routing using a similar method then the batch filtering track. The new representing profile obtained from the sample documents is selected as routing profile, and applied on the test documents. The final output is the top 1000 ranked documents for each topic.
Table 3 shows the routing results at average uninterpolated precision for all topics.

TREC Routing for topics 101-150			
= *max*	≥ *median*	< *median*	*AvgP*
13	45	5	0.369

TREC Routing for topics 151-200			
= *max*	≥ *median*	< *median*	*AvgP*
11	42	8	0.004

Table 3: Comparative routing results

3 Conclusion

We described in this paper a learning and threshold updating method for information filtering. Adaptive learning is based on equation system resolution under constraints, a gradient propagation formula uses the system solution to improve the user profile representation. The threshold updating is done independently from learning, it controls perfectly the random variation of *rsv* values affected to incoming documents.
We have presented our experiments for TREC2002 who are focused on the Filtering (adaptive, batch and routing) tracks.

References

[1] M. Boughanem, C. Chrisment, M. Tmar, *Mercure and MercureFiltre applied for Web and Filtering tasks at TREC-10.* Proceedings of TREC-10, 2001.

[2] M. Boughanem and M.Tmar, *Incremental profile adaptive filtering: profile learning and threshold calibration,* Proceedings of 17th ACM Symposium on Applied Computing (SAC), pages 640-644, 2002.

[3] M. Mostafa, *An algorithm for suffix stripping.* Program, 14(3), pages 130-137, 1980.

[4] G. Saporta, *Probabilits, analyse des donnes et statistiques.* ed. Technip, 1996.

[5] S. Walker, S. E. Robertson, M. Boughanem, G. J. F. Jones, K. Sparck Jones, *Okapi/Keenbow at TREC-6 automatic and ad hoc. VLC. routing, filtering and QSDR.* Proceedings of TREC-6, 1997.

IRIT at TREC'2002: Web Track

A. Benammar, M. Boughanem, G. Hubert, C. Laffaire, J. Mothe

IRIT-SIG
Campus Univ. Toulouse III
118, Route de Narbonne
F-31062 Toulouse Cedex 4
Email : trec@irit.fr

1 Summary

The tests we performed for TREC'2002 web track focus on the web distillation part. The aim of our participation is to experiment our method for topic distillation combined with a new version of our system Mercure and to validate our system on a large collection of web pages: 18 Go of data.

This year, three runs were submitted to NIST.

2 Mercure model

Mercure is an information retrieval system based on a connexionist approach and modeled by a multi-layered network. The network is composed of a query layer (set of query terms), a term layer (representing the indexing terms) and a document layer [Boughanem99].

Mercure includes the implementation of a retrieval process based on spreading activation forward and backward through the weighted links. Queries and documents can be used either as inputs or outputs. The links between layers are symmetric and their weights are based on *tf-idf* measure inspired from OKAPI [Robertson00] and SMART term weighting.

- the query-term links (at stage s) are weighted as follows:

$$q_{ui}^{(s)} = \begin{cases} \dfrac{nq_u \times qtf_{ui}}{nq_u - qtf_{ui}} & if\,(nq_u > qtf_{ui}) \\ qtf_{ui} \quad otherwise \end{cases} \tag{1}$$

Where:

□ $q_{ui}^{(s)}$: the weight of the term t_i in the query u at the stage s,

□ qtf_{ui}: the query term frequency of t_i in the query u,

□ nq_u: number of terms in the query u,

- the term-document link weights are expressed by:

$$d_{ij} = \frac{tf_{ij} \times \left(h_1 + h_2 \times \log\left(\dfrac{N}{n_i}\right)\right)}{h_3 + h_4 \times \dfrac{dl_j}{\Delta d} + h_5 \times tf_{ij}} \tag{2}$$

Where:

□ d_{ij}: term-document weight of term t_i and document d_j,

- tf_{ij}: the term frequency of t_i in the document d_j,
- N: the total number of documents,
- n_i: the number of documents containing term t_i,
- h_1, h_2, h_3, h_4 and h_5: constant parameters,
- Δd : average document length.

3 Web Track Experiment

3.1 Indexing methodology

We validate our indexing scripts on the GOV collection. Our scripts have been modified to optimize the build of the dictionary: we obtain now a gain of 50% for the speed-up of the execution.

The queries used for the runs were indexed only with the title field.

3.2 Web methodology

Once the GOV collection has been indexed, three runs were performed and submitted to NIST. The first one, Mercah, is based on a sample search issued from an evolution of our system Mercure. This run is an ad-hoc retrieval. It has been performed with no relevance feedback and no query expansion

The second one, Mercure, is based on the analysis of the domain of each document in the aim of finding the hit page.

The third one, MercureLynx, was carried out using the results of Mercah ad-hoc search, and then the list of selected documents is re-ranked using the link analysis method we propose. The approach used for this run is described in section 4.

3.3 Web Distillation

Our algorithm for the web distillation experiments is derived from the HITS algorithm proposed by Kleinberg for ranking search engine results [Kleinberg98]. We extend the HITS algorithm to exploit not only links but also the document contents in order to re-rank the document list retrieved by Mercure search engine. The proposed algorithm is composed of the following steps:

1. Neighborhood graph construction: a neighborhood graph is a directed graph consisting of a set of nodes (documents) and directed edges (hypertext links) between nodes. For a given subset of documents, we construct a neighborhood graph containing all the links between documents. The neighborhood is composed only of the documents that appear in the retrieved document set. we consider the 1000 top ranked documents in the first retrieved result.
2. Neighborhood graph analysis: this analysis is based simultaneously on the document links and on the contents. HITS algorithm does not weight the edges of the graph. However, in their experiments, Bharat and Henzinger [Henzinger98] have shown that edge weights improve the precision. Indeed, edge weights reduce the influence of documents that are all contained in one host. In our approach, we define a weighting method that depends on:

- the link typology: In a neighborhood graph, there are two kinds of hypertext links: organizational and navigational. An organizational link relates two documents belonging to the same host (WWW domain) and a navigational link relates two documents belonging to different hosts. In the experiment presented in this paper, we do not consider the organizational links. However, in further experiments, we will consider both organizational and navigational links but giving less importance to organizational ones.
- the link relevance:

The relevance weight of the link from d_1 to d_2 ($wl_{d_1 \to d_2}$) is calculated as follows:

$$wl_{d_1 \to d_2} = \beta \times R(d_1, d_2) \qquad (3)$$

Where:

- d_1 and d_2: are documents from the neighborhood graph,
- β: is a parameter used to weight differently organizational and navigational links. β is equal to 1 for navigational links, and to 0 for organizational links.
- $R(d_1, d_2) = Similarity \left(\overrightarrow{d_1}, \overrightarrow{Q}\right) \times Similarity \left(\overrightarrow{d_2}. \overrightarrow{Q}\right)$

We use inner product normalization when evaluating the similarity between a document d_i and the query Q:

$$Similarity \left(\overrightarrow{d_i}, \overrightarrow{Q}\right) = \sum_{j=1}^{t} \left(w_{jq} \times w_{ji}\right) \qquad (4)$$

Where w_{ji} (respectively w_{jq}) corresponds to the *tf-idf* value of the j^{th} term in the document d_i (respectively in the query Q).

4 Results

Table 1 describes the results obtained at TREC'2002 Topic distillation task for officials runs.

Precisions:	*Mercah*	*Mercure*	*MercureLynx*
A 5 documents:	0.2449	0.2041	0.1184
A 10 documents	0.2163	0.1429	0.1082
A 15 documents	0.2041	0.0966	0.0898
A 20 documents	0.1765	0.0724	0.0776
A 30 documents:	0.1463	0.0483	0.0728
A 100 documents	0.0898	0.0145	0.0429
A 200 documents	0.0661	0.0072	0.0293
A 500 documents	0.0356	0.0029	0.0230
A 1000 documents	0.0203	0.0014	0.0203
Exact:	0.1984	0.0575	0.0646

Table 1

It is significant that our best performing run is Mercah which is an ad-hoc retrieval run. We identify two main reasons of these results. First, it confirms that the GOV collection is well indexed and that our scripts are performing. Secondly, it means that best resources for topic distillation such as hit pages can be found with an ad-hoc research engine and are ranked in the top ten documents retrieved. It also confirms that the weight assign to terms is well estimated.

About the run Mercure, it is not surprising that results we obtained are not significant because we retrieved only 415 documents for the 50 queries. Our precision at 5 documents and precision at 10 documents are good because we retrieved less than 10 documents per query. So we can think, we retrieved 10 good resources. Of course, all other precisions are not significant because they are calculated with only 10 documents per topic. It is due to our algorithm which extracts the domain for each document and retrieved only one page per domain. For each query, we found approximately 10 different domains.

The run MercureLynx are not significant. In fact, the proposed algorithm is not perfectly personalized to the topic distillation task. Indeed, the goal of the topic distillation is to retrieve the most relevant resource. However, the aim of our algorithm is to enhance the precision values of the final ranking by including more relevant documents in the first levels of the final result.

However, the experiments have shown the importance of the link analysis precisely by combining the content and hypertext analyses.

5 Conclusion

The results obtained this year in TREC 2002 Web Track show that our system Mercure is able to obtain good results with large collection of data.

It also shows that an ad-hoc research can obtain good results even if it is not specially developed for topic distillation task, so we will work next year to ameliorate our system with evolution such as document structure and query expansion.

References

[Boughanem99] Boughanem M., Chrisment C., Soule-Dupuy C., *Query modification based on relevance back-propagation in ad-hoc environment*, Information Processing and management, 35 (1999), pages 121-139, 1999.

[Henzinger98] Henzinger M., Bharat K., *Improved algorithms for topic distillation in a hyperlinked environment,* 21st International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 101-111, Melbourne, 1998.

[Kleinberg98] Kleinberg J. M., *Authoriatives sources in a hyperlinked environment*, Journal of the ACM, Volume 46, Number 5, pages 604-632, 1998.

[Robertson 00] Robertson S. E., Walker S., *Okapi/Keenbow at TREC-8*, In Proceedings of the TREC-8 Conference, National Institute of Standards and Technology, pages 151-161, 2000.

Mining Knowledge from Repeated Co-occurrences: DIOGENE at TREC-2002

Bernardo Magnini, Matteo Negri, Roberto Prevete and Hristo Tanev
ITC-Irst, Centro per la Ricerca Scientifica e Tecnologica
Via Sommarive, 38050 Povo (TN), Italy
{magnini,negri,prevete,tanev}@itc.it

Abstract

This paper presents a new version of the DIOGENE Question Answering (QA) system developed at ITC-Irst. With respect to our first participation to the TREC QA main task (TREC-2001), the system presents both improvements and extensions. On one hand, significant improvements rely on the substitution of basic components (*e.g.* the search engine and the tool in charge of the named entities recognition) with new modules that enhanced the overall system's performance. On the other hand, an effective extension of DIOGENE is represented by the introduction of a module for the automatic assessment of the candidate answers' quality. All the variations with respect to the first version of the system as well as the results obtained at the TREC-2002 QA main task are presented and discussed in the paper.

1 Introduction

The new version of the DIOGENE QA system described in this paper is based on last year's version (Magnini et al., 2001), focusing on two main directions: the improvement of its basic components and the extension of the original architecture.

First, the architecture of the system has been improved by substituting part of its modules and algorithms with more suitable and reliable solutions. Since an analysis of the information flow throughout the process indicated the *search component* and the *answer extraction component* were the main error sources in our previous participation to the competition, most of the improvements concern these aspects of the architecture. In particular, a new search engine, new document indexing techniques, new query formulation criteria and a new module for named entities recognition have been adopted.

Second, the system has been extended by adding a module for a fast and totally automatic evaluation of candidate answer strings (Magnini et al., 2002a). The main reason behind the necessity of providing the system with an answer validation component concerns the difficulty of picking up from a document the "exact answer" required by the TREC-2002 main task guidelines. Moreover, one of the lessons learned after our first participation to the TREC QA main task was the importance of a reliable distinction between possible correct answers and the huge quantity of spurious material retrieved by the search engine. As an example, given the question "*Who is Tom Cruise married to?*" and the text snippet "*Married actors Tom Cruise and Nicole Kidman play Dr. William and Alice Harford, a wealthy New York couple who think their eight-year marriage is very, very good.*", we had to deal with the difficulty of understanding who is the real Tom Cruise's wife and select the exact answer "Nicole Kidman" among the candidates. Our approach to automatic answer validation relies on discovering relations between a question and the answer candidates by mining the Web or a large text corpus for their co-occurrence tendency. The underlying hypothesis is that the number of these co-occurrences can be considered a significant clue to the validity of the answer. As a consequence, this information can be effectively used to rank the huge amount of candidate answers that our QA system is often required to deal with. Considering the above example, the introduction of the automatic answer validation component met the specific need of providing DIOGENE with an effective way of filtering out the improper candidate "Alice Harford" and choosing the best exact answer within the document retrieved by the search engine.

Since the overall system's architecture is slightly similar to the one described in (Magnini et al., 2001), this paper will be mainly focused on the description of the novelties of this year's version of DIOGENE.

After a short description of the *question processing component* in Section 2, the main features of our new *search component* will be presented in Section 3. Then, the *answer extraction component* will be thoroughly analyzed in Section 4, with a particular emphasis on the details of our approach to automatic answer validation (Section 4.2). Section 5 will conclude the paper illustrating the results achieved by our system at TREC-2002, providing a preliminary error analysis and some final remarks about strengths and weaknesses of DIOGENE.

2 Question Processing Component

Apart from the introduction of the answer validation component, the architecture of DIOGENE has not changed. The system still relies on three basic components (see Figure 1), namely the *question processing* component (in charge of the linguistic analysis of input questions), the *search component* (which performs the query composition and the document retrieval), and the *answer extraction* component (which extracts the final answer from the retrieved text passages).

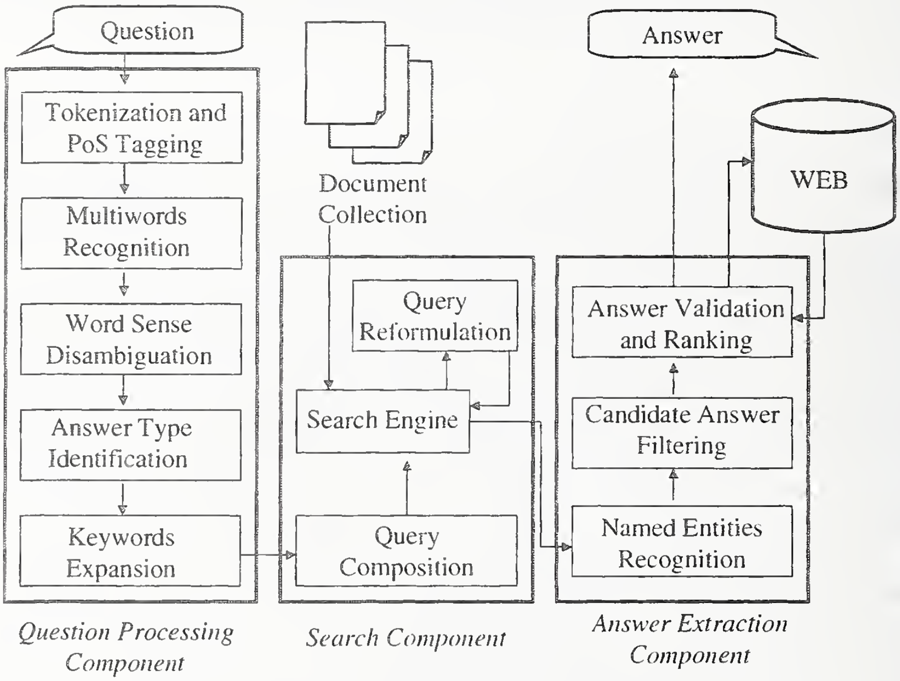

Question
Tokenization and PoS Tagging
Multiwords Recognition
Word Sense Disambiguation
Answer Type Identification
Keywords Expansion
Question Processing Component
Document Collection
Query Reformulation
Search Engine
Query Composition
Search Component
Answer
WEB
Answer Validation and Ranking
Candidate Answer Filtering
Named Entities Recognition
Answer Extraction Component

Figure 1. DIOGENE Architecture.

During the *question processing* phase, the linguistic analysis of the input is performed sequentially by the following modules:

- ***Tokenization and PoS tagging***. First, the question is tokenized and words are disambiguated with respect to their lexical category by means of the Treetagger (Schmid, 1994), a statistical Part of Speech tagger developed at the University of Stuttgart.
- ***Multiwords recognition***. About five thousand multiwords (i.e. collocations, compounds and complex terms) have been automatically extracted from WORDNET (Fellbaum, 1998) and are recognized by means of pattern matching rules.
- ***Word sense disambiguation***. The identification of the correct sense of the question terms is necessary to expand the search query with synonyms of that words without the risk of introducing

disturbing elements. In order to provide correct synonyms for a reasonable query expansion, question words are disambiguated with respect to their senses.

- ***Answer type identification***. The answer type for a question represents the entity to be searched as an answer: this information is used to select the correct answer to an input question within the documents retrieved by the search engine. In particular, knowing the category of the entity we are looking for (e.g. PERSON, LOCATION, DATE, etc.) we can determine if any "candidate answer" found in a document is an appropriate instantiation of that category. Our answer type identification module relies on a manually defined taxonomy of answer types (e.g. "PERSON", "LOCATION", "ORGANIZATION", "DATE", etc.) and a set of approximately 240 rules that check different features of the input question. Answer type identification is the aspect of the question processing component that presents the most significant improvements with respect to last year's version of DIOGENE. In order to enlarge the coverage of possible types of questions and refine the answer type taxonomy, more than 100 rules have been added to the original module used for our first participation inn the TREC competition.
- ***Keyword expansion***. At the end of the linguistic processing of the question, a stop words filter is applied that cuts off both non-content words and non-relevant content words. The remaining words (we call them "basic keywords") are then passed to an expansion phase which considers both morphological derivations and synonyms.

3 Search Component

The search component first composes the question keywords and their lexical expansions in a Boolean query, then performs document retrieval from the AQUAINT text collection. In order to overcome the main difficulties encountered last year using the Zprise search engine provided by NIST (i.e. the lack of Boolean expression support, as well as the necessity of considering a maximum of ten keywords per query and retrieving a maximum of 10 documents per question in order to bring the processing time under control), this year we adopted Managing Gigabytes (MG) (Witten et al., 1999) as a new search engine. MG is an open-source indexing and retrieval system for text, images, and textual images covered by a GNU public license and available via ftp from *http://www.cs.mu.oz.au/mg/*.

Besides the speed of the document retrieval, the advantages derived from using MG are twofold. First, it allows for the customization of the indexing procedure. As a consequence, we opted to index the AQUAINT collection at the paragraph level, using the paragraph markers provided in the SGML format of the documents. This way, although no proximity operator (e.g. the "NEAR" operator provided by AltaVista) is implemented in MG, the paragraph index makes the "AND" Boolean operator perform proximity search. In order to divide very long paragraphs into short passages, we set 20 text lines as the limit for paragraphs' length. This new indexing criterion allowed us to avoid the huge quantity of errors related to the paragraph filtering techniques used in last year's version of DIOGENE.

The other advantage derived from using MG concerns the possibility of performing Boolean queries, thus obtaining more control over the terms that must be present in the retrieved documents. Using the Boolean query mode, at the first step of the search phase all the basic keywords are connected in a complex "AND" clause, where the term variants (morphological derivations and synonyms) are combined in an "OR" clause. As an example, the question "*When did Titanic sink?*" is transformed into:

[**Titanic** AND (**sink** OR **sank** OR **sunk**)]

However, Boolean queries often tend to return too many or too few documents. To cope with this problem, we implemented a feedback loop which starts with a query containing all the basic keywords and gradually simplifies it by ignoring some of them. Several heuristics are used by the algorithm. For example, a word is removed if the resulting query does not produce more than a fixed number of hits (this probably means that the word is significant). Other heuristics consider the capitalization of the query terms, their part of speech, their position in the question, WordNet class, etc. (see Magnini et al., 2002b). The algorithm stops when a maximum of 150 text paragraphs has been collected or a certain percentage of the question terms has been cut off. This way, the searching algorithm builds a set of the most significant words and narrows it until enough documents are retrieved. The efficiency of these kinds of feedback loops has been recently pointed out by (Harabagiu et al., 2001).

4 Answer Extraction Component

The answer extraction component is the other aspect of the architecture that has been considerably updated. As stated before (Section 1), most of the efforts were dedicated to the substitution of the named entities recognition module (the performance analysis of the tool used in the last years' version of the system showed a 60% error rate), and the extension of DIOGENE with a module for the automatic evaluation and ranking of the answer candidates. Both of the new modules are described in the following sections.

4.1 Named Entities Recognition

Once the relevant paragraphs have been retrieved, the named entities recognition module is in charge of identifying within these text portions all the entities that match the answer type category (e.g. PERSON, ORGANIZATION, LOCATION, MEASURE, etc.). The task is performed by a rule-based named entities recognition system for the English written language developed at ITC-Irst (Magnini et al. 2002c). The core of the system relies on the combination of a set of language dependent rules with a set of predicates, defined on the WORDNET hierarchy for the identification of both proper names (i.e. person, location and organization names, such as "Galileo Galilei", "Rome", and "Bundesbank") and *trigger words* (i.e. predicates and constructions typically associated with named entities, such as "astronomer", "capital", and "bank").

The process of recognition and identification of the named entities present in a text is carried out in three phases. The first phase (*preprocessing*) performs tokenization, PoS-tagging, and multiwords recognition in the input text. In the second phase, a set of approximately 200 *basic rules* is used for finding and marking with SGML tags all the possible named entities present in the text (e.g. <MEASURE><CARDINAL>*200*<\CARDINAL> *miles*<\MEASURE> *from* <LOCATION>*New York*<\LOCATION>). Finally, a set of higher level *composition rules* is used to remove inclusions and overlaps among tags (e.g. <MEASURE>*200 miles*<\MEASURE> *from* <LOCATION>*New York*<\LOCATION>) as well as for co-reference resolution.

The system has been tested using the test corpora and the scoring software provided in the framework of the DARPA/NIST HUB4 evaluation exercise (Chinchor et al., 1998). Results achieved over a 365Kb test corpus of newswire texts vary among categories, ranging from an F-Measure score of 71% for the category MEASURE, to 96.5% for the category DATE.

4.2 Answer Validation

The answer validation module is in charge of evaluating and scoring a maximum of 40 answer candidates per question in order to find the exact answer required as the final output. The top 40 answer candidates are selected, among the named entities matching the answer type category, on the basis of their distance from the basic keywords and their frequency in the paragraphs retrieved by the search engine.

The basic idea behind our approach to answer validation is to identify semantic relations between concepts by mining for their tendency to co-occur in a large document collection. In this framework, considering the Web as the largest open domain text corpus containing information about almost all the different areas of the human knowledge, all the required information about the relation (if exists) between a question *q* and an answer *a* can be automatically acquired on the fly by exploiting Web data redundancy.

In particular, given a question q and an answer a, it is possible to combine them in a set of *validation statements* whose truthfulness is equivalent to the degree of relevance of a with respect to q. For instance, given the question "*What is the capital of the USA?*", the problem of validating the answer "*Washington*" is equivalent to estimating the truthfulness of the validation statement "*The capital of the USA is Washington*". Therefore, the answer validation task could be reformulated as a problem of statement reliability. There are two issues to be addressed in order to make this intuition effective. First, the idea of a validation statement is still insufficient to catch the richness of implicit knowledge that may connect an answer to a question. Our solution to this problem relies on the definition of the more flexible idea of a *validation pattern*, in which the question and answer keywords co-occur closely. Second, we need an effective and efficient way to check the reliability of a validation pattern. With regard to this issue, we propose two solutions relying on a statistical count of Web searches and on document content analysis respectively. A detailed description and a comparison between the experimental results achieved by the two approaches is presented in (Magnini et al., 2002a).

With reference to the above considerations, given a question-answer pair [*q,a*] we propose the following generic scheme for answer validation. Both the statistical and the content-based approach perform four basic steps:

1) Compute the set of representative keywords *Kq* and *Ka* both from q and from a. This step is carried out using linguistic techniques, such as answer type identification (from the question) and named entities recognition (from the answer);
2) From the extracted keywords construct the validation pattern for the pair [*q,a*];
3) Submit the validation pattern to a search engine;
4) Estimate an *Answer Relevance Score (ARS)* considering the results returned by the search engine.

The retrieval on the Web is delegated to a publically available search engine (e.g. AltaVista or Google). The post-processing of the results is performed by HTML parsing procedures and simple functions which calculate the *ARS* for every [*q, a*] pair by analyzing the results pages returned by the search engine. The two algorithms for automatic answer validation diverge in the methodology for the *ARS* calculation as well as for the search engine used; nevertheless, in both cases Web documents are not downloaded, thus making the algorithms rather efficient.

Statistical approach. The pure statistical approach makes use of the AltaVista search engine (http://www.altavista.com), exploiting the proximity operator "*NEAR*" to retrieve only Web documents where the answer and the question keywords co-occur. The *ARS* is then calculated on the basis of the number of retrieved pages by means of a statistical co-occurrence metric called *corrected conditional probability* (Magnini et al., 2002b). The formula we used is the following:

$$ARS(a) = \frac{P(Ka \mid Kq)}{P(Ka)^{2/3}} = \frac{hits(Ka\ NEAR\ Kq)}{hits(Kq) * hits(Ka)^{2/3}} * |EnglishPages|$$

where:

- *hits(Ka NEAR Kq)* is the number of English-language pages returned by AltaVista, where the answer keywords (*Ka*) and the question keywords (*Kq*) are in distance of no more than 10 words of each other;
- *hits(Kq)* and *hits(Ka)* are the number of English-language pages where *Kq* and *Ka* occur respectively;
- |*EnglishPages*| is the number of English pages, indexed by AltaVista.

This formula can be viewed as a modification of the Pointwise Mutual Information formula, a widely used measure that was first introduced for identifying lexical relationships (in this case the co-occurrence of *Kq* and *Ka*).

Content-based approach. The content-based approach makes use of Google (http://www.google.com), taking advantage of the text passages (i.e. snippets) returned by the search engine as output of a Web search. Using the fact that Google ranks higher the documents where the query terms co-occur close to each other, the *ARS* is calculated considering the presence of relevant keywords within the top 100 retrieved snippets. The underlying assumption is that the closer the distance between the candidate answer *a* and the question keywords within these text passages, the stronger their relation is.
Every appearance of the candidate answer *a* in a snippet is evaluated by calculating a *co-occurrence weight*, as the number of the question keywords and their distance from *a*. If we have co-occurrence of the answer *a* and a set of question keywords QK = $\{qk_1, qk_2, ...\}$ the co-occurrence weight CW(*a*,QK) is calculated by means of the following formula:

$$\mathrm{CW}(a,\mathrm{QK}) = \prod_i w(qk_i)^{(\|qk_i,a\|+1)^{-1}}$$

Where:

- $w(qk_i)$ is the weight of the question keyword qk_i. In general, $w(qk_i)$ can be calculated from the keyword frequency. However in the current implementation of the algorithm we used equal weights for all the words.
- $\| qk_i, a \|$ denotes the distance between the answer *a* and the closest appearance of qk_i.

If we denote with S_a the set of the top 100 text snippets, the *ARS* is calculated through the formula:

$$ARS = \sum_{QK \in S_a} CW(a, QK)$$

This formula gives high preference to the answers which occur close the question keywords.

As stated before, validation patterns capture the relation (if one exists) between a question and an answer through simple co-occurrence mining. However, an additional pattern-based approach exploiting the information conveyed by the presence within the Web documents of explicit validation statements (e.g. phrase patterns such as "*The capital of the USA is Washington*") has been partially explored. In this version of DIOGENE, the use of a kind of phrase pattern slightly similar to the ones described in (Subbotin and Subbotin, 2001) has also been tested for the simplest possible variant of the "*Where is*" questions. In particular, if the question is of the type "*Where is <NP>?*" (where <NP> stands for a simple noun phrase without attached prepositional phrases), we search the Web for the phrase pattern ["<NP> in *a*"]. The number of hits produced by the search is then used to increase the *ARS*.
This solution proved to be rather effective and, in some cases, allowed DIOGENE to avoid errors produced by the simple co-occurrence mining techniques. As an example, given the question "*Where is the Orinoco River?*" and the two answer candidates "*Amazon*" and "*Venezuela*" both the statistical and the content-based approaches gave preference to "*Amazon*" as the best final answer. However, a Web search with the string ["Orinoco River in Amazon"] did not find any documents, while the string ["Orinoco River in Venezuela"] returned respectively 322 hits using Google and 104 using AltaVista, thus confirming that the location of the Orinoco River is Venezuela. In this case, the *ARS* obtained considering the presence of the phrase pattern ["Orinoco River in Venezuela"] into the Web documents led DIOGENE to the correct answer.
In general, the exploitation of different levels of patterns, ranging from the more general validation patterns to the more specific phrase patterns is of great interest, and seems to be a simple and powerful

instrument for answer extraction and validation. Exploitation of such patterns requires a very detailed question taxonomy and the development of machine learning techniques for their automatic acquisition.

4.3 Answer ranking

This year the ranking of the answers to the 500 questions of the QA main task was of great importance for the final score. In fact, a measure that is an analogue to the document retrieval's uninterpolated average precision was used to score the runs. The *Confidence-Weighted Score* (*CWS*) formula gives higher weights to the answers for which systems are more confident (i.e. answers with a higher rank in the run submissions), thus penalizing systems unable to accomplish a reliable calculation of the answers' confidence level.

As stated before, DIOGENE exploits the results of the answer validation also at the answer ranking phase. The task is accomplished combining the ARS with a *Question Type Reliability* (*QTR*) coefficient which indicates how reliable the ARS is with respect to a given question type. As an example, the QTR associated with the questions asking for a PERSON is 1, while the QTR for ORGANIZATION questions is 0.5 and for LOCATION questions is 0.75. The QTR coefficient for the different question types was computed considering the results of the answer validation experiments described in (Magnini et al 2002a). Given the QTR and the ARS, the confidence level (CFL) is calculated by the following formula:

if the answer is not NIL , $CFL = QTR * ARS$
if the answer is NIL , $CFL = QTR * 0.1$

where NIL means that no answer was found in the target corpus.

Since the ARS is usually much higher than 1, this formula gives a lower preference to the NIL answers, pushing them toward the end of the run submission (in our submission, the first NIL answer was ranked 406th). This is motivated by the fact that, as we process only a maximum of 150 paragraphs per question, there is no certainty about the correctness of the NIL answers. As a consequence, giving the NIL answers a lower rank reduces the impact of errors caused by the possible false negatives.

# Answers	Right	Unsupported	Inexact	Wrong
1 to 100	73	7	8	12
100 to 200	44	10	5	41
200 to 300	39	2	3	56
300 to 400	20	4	1	75
400 to 500	16	1	0	85
Total	192	24	17	267

Table 1. Distribution of Right, Unsupported, Inexact, and Wrong answers.

Table 1 shows how correct, unsupported, inexact and wrong answers have been ranked by DIOGENE in the best of the three runs submitted. Results confirm that our answer ranking technique performed well, producing an output list where most of the correct answers are distributed at the top (73% of the top ranked 100 answers are correct).

5 Results and Discussion

DIOGENE's performance has been evaluated over three runs submitted to the TREC-2002 QA main task (see Table 2). The three answer lists have been produced using the same architecture, simply by varying the answer validation algorithm in order to test the impact of the different approaches. The best classified run (with a confidence-weighted score of 0.589, around 6% above the other two) was obtained using the content-based answer validation approach, while the second classified resulted from the combination of the statistical and the content-based techniques, and the third resulted from the application of the statistical approach.

Run	Right	Unsupported	Inexact	Wrong	CWS
IRST02D1	192	24	17	267	0.589
IRST02D3	177	23	16	284	0.533
IRST02D2	173	19	14	294	0.520

Table 2. ITC-Irst at TREC-2002.

In order to evaluate strengths and weaknesses of DIOGENE, an error analysis was carried out considering the first 100 questions where the system failed. Also this year, most of the errors (40%) came from incorrect document retrieval. An in depth analysis of the search phase results revealed two main sources of errors. First, the stemming algorithm used by MG leads to the retrieval of many irrelevant paragraphs. Second, many errors are due to the difficulty of dealing with the variety of lexical formulations of an answer with respect to a question. The solution to this problem requires the development of intelligent query formulation criteria, going beyond the simple algorithms for keyword extraction and query expansion with synonyms and morphological derivations. For instance, reliable query formulation criteria should consider the relation between the question semantics and the possible transformations of its surface form. In spite of the remarkable improvements brought to the overall system's performance, answer validation is the reason for 38% of the errors. Most of these errors came from the fact that our approach measures the co-occurrence between the entities and does not consider the semantic relation which is the origin of that co-occurrence. As an example, given the question "*What is Buzz Aldrin's real first name*?", our answer validation component returned "*Neil Armstrong*" (the person name most frequently co-occurring with the question keywords) instead of the correct answer "*Edwin*". The answer candidates extraction produced 19% of the errors. In some cases this is due to the fact that DIOGENE is still unable to determine the correct answer type for some classes of questions (i.e. "Why" and "How" questions, such as "*Why does the moon turn orange*?", and "*How did Mahatma Gandhi die*?"), thus providing a huge number of irrelevant answer candidates. In some other cases the correct candidates have been discarded because of their distance from the query keywords within the retrieved paragraphs or because of errors in the named entity recognition phase. Also this year, the answer type extraction module performed well, with an error rate of only 3% due to PoS-tagging and disambiguation errors.

References

Chinchor, N., Robinson, P., Brown, E.: Hub-4 Named Entity Task Definition (version 4.8). Technical Report, SAIC. *http://www.nist.gov/speech/hub4_98.*

Fellbaum, C.: WORDNET, An Electronic Lexical Database. The MIT Press (1998).

Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Girjiu, R., Rus, V., Morarescu, P.: The Role of Lexico-Semantic Feedback in Open-Domain Question-Answering. Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001). Toulouse, France (2001).

Magnini, B., Negri, M., Prevete, R., Tanev, H.: Multilingual Question/Answering: the DIOGENE System. Proceedings of the Tenth Text Retrieval Conference (TREC-10), Gaithersburg, MD. (2001).

Magnini, B., Negri, M., Prevete, R., Tanev, H.: Comparing Statistical and Content-Based Techniques for Answer Validation on the Web. Proceedings of the VIII Convegno AI*IA, Siena, Italy, (2002a).

Magnini, B., Negri, M., Prevete, R., Tanev, H.: Is It the Right Answer? Exploiting Web Redundancy for Answer Validation. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-2002), Philadelphia, PA. (2002b).

Magnini, B., Negri, M., Prevete, R., Tanev, H.: A WordNet-Based Approach to Named Entities Recognition. Proceedings of SemaNet02, COLING Workshop on Building and Using Semantic Networks, Taipei, Taiwan, (2002c).

Moldovan D., Harabagiu S., Pasca M., Girju R., Goodrum R., Rus V.: The Structure and Performance of an Open-Domain Question Answering System. Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL), Hong Kong, (2000).

Schmid, H.: Probabilistic Part-Of-Speech Tagging Using Decision Trees. Proceedings of the International Conference on New Methods in Language Processing (1994).

Subbotin, M., Subbotin, S.: Patterns of Potential Answer Expressions as Clues to the Right Answers. Proceedings of the Tenth Text Retrieval Conference (TREC-10), Gaithersburg, MD. (2001).

Witten, I. H., Moffat, A., Bell T.: Managing Gigabytes: Compressing and Indexing Documents and Images (second ed.), Morgan Kaufmann Publishers, New York (1999)

JHU/APL at TREC 2002: Experiments in Filtering and Arabic Retrieval

Paul McNamee, Christine Piatko, and James Mayfield
Research and Technology Development Center
Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road, Laurel, Maryland 20723-6099 USA
{mcnamee, piatko, mayfield}@jhuapl.edu

Overview

The Johns Hopkins University Applied Physics Laboratory (JHU/APL) participated in two tracks at this year's conference. We participated in the filtering track, again addressing the batch and routing subtasks, as well as the adaptive task for the first time. We also continued experiments in Arabic retrieval, emphasizing language-neutral approaches.

For ranked retrieval, we relied on a statistical language model to compute query/document similarity values. Hiemstra and de Vries describe such a linguistically motivated probabilistic model and explain how it relates to both the Boolean and vector space models [4]. The model has also been cast as a rudimentary Hidden Markov Model [13]. Although the model does not explicitly incorporate inverse document frequency, it does favor documents that contain more of the rare query terms. The similarity measure can be computed as

$$Sim(q,d) = \prod_{t \in q} \left(\alpha \cdot f(t,d) + (1-\alpha) \cdot df(t)\right)^{f(t,q)}$$

Equation 1. Similarity calculation.

where α is the probability that a query word is generated by a document-specific model, and *(1- α)* is the probability that it is generated by a generic language model. *df(t)* denotes the mean relative document frequency of term *t*. In our experiments an α of between 0.15 and 0.3 has worked well, but performance is fairly insensitive to the precise value used.

For text classification problems we used Support Vector Machines (SVMs), which efficiently perform binary classification tasks. We applied SVMs to this year's Filtering tasks; however, some of our routing runs were based on statistical language models instead.

Filtering Track

We participated in the routing, batch and adaptive tasks of the filtering track.

Filtering Approach Background

We continued to investigate the application of Support Vector Machines (SVMs) to filtering tasks. SVMs are used to create classifiers from a set of labeled training data, finding a hyperplane (possibly in a transformed space) to separate positive examples from negative examples. This hyperplane is chosen to maximize the margin (or distance) to the training points. The promise of large margin classification is that it does not overfit the training data and generalizes well to test data of similar distribution. See Hearst [3] for a general discussion of SVMs. We used the SVM-light package (version 3.50, by Thorsten Joachims [15]) to create classifiers based on the training data for classification of the test data, and wrote a JNI interface to SVM-light to support filtering with our HAIRCUT system. All runs used stem indices using a derivative version of the SMART stemmer.

We slightly reduced the term space to create test and training document vectors. Terms were selected using the top stems by document frequency in the training set. (Exact numbers of stems differed for different runs, noted per task in descriptions below.) Stopwords were not removed. We used tf/idf weighted vectors for each document. IDF values were based on training index statistics. Vectors were normalized to unit length. Given *n* positive training documents for a topic, we chose either all other training qrels documents as (presumed) negative examples, or randomly sampled (number of known positive examples) * *NegativeToPositiveRatio* presumed negative examples from the training index, throwing away any that were actually positive. (The use of all negatives or a particular ratio is noted below.) We trained linear SVMs, weighting positive and negative training examples equally (*-j 1* flag in SVM-light).

Filtering Training

Over the course of the year we had performed several experiments using the Reuters Corpus [18] and topics from TREC 2001. Based on track guidelines, we wanted to establish various parameters necessary for our system based on alternative data. We chose to reset these based on performance on Financial Times data that had been used in the TREC-8 Filtering Track. We did this in a straightforward way for the routing and batch training and test sets. However, no training documents had been used for the adaptive task in TREC-8, so for this training we randomly selected three relevant training documents from the batch FT training qrels for each topic.

Routing Task

We submitted two official runs for routing. We submitted an SVM based run *apl11Fsvm* and a rank-based merge run *apl11Frm*, the merge of the SVM run with an unofficial score-based run *apl11Frs*.

Our statistical language model-based run *apl11Frs* used simulated routing (using a modified version of our HAIRCUT system to score indexed test documents using training index statistics). We formed queries using 60 terms per topic that were selected from the positive qrels training documents. Term selection was accomplished using mutual information based difference statistics with respect to the training documents.

For the SVM routing run *apl11Fsvm*, we used the top 12000 features ranked by document frequency and 10 times as many negative documents as known positive examples to train an SVM. We kept track of the top 1000 documents for a topic in a heap based on the SVM score.

	Avg. prec.	# terms	# bests	# ≥ median (50 topics)
apl11Frm	0.330	12000	4	43
apl11Fsvm	0.218	12000	1	24
apl11Frs	0.364	60	Score run	
apl11Frsvm2	0.364	40000	SVM run	
apl11Frm2	0.412	40000	Merge run	

Table 1. APL Routing Results, Assessor topics. Highlighted rows indicate unofficial runs.

	Avg. prec.	# bests	# ≥ median (50 topics)
apl11Frm	0.042	5	37
apl11Frsvm	0.043	4	35
apl11Frs	0.035	Score run	
apl11Frsvm2	0.041	SVM run	
apl11Frm2	0.045	Merge run	

Table 2. APL Routing Results, Intersection topics. Highlighted rows indicate unofficial runs.

In subsequent analysis of our results we realized that we submitted the wrong SVM-based routing run. We had intended to submit the SVM run *apl11Frsvm2* based on 40000 df terms (the best of numbers to use on FT data). This run does well, and makes an excellent run *(apl11Frm2)* when rank-merged with the score-based run.

Batch Filtering Task

We used the score of the test document from the topic-specific SVM to decide whether to return a document as possibly relevant. We used cross-validation for threshold selection. We applied *n*-fold cross-validation on training data to find the best threshold per topic for the given score function being optimized.

Lewis had applied exhaustive leave-one out to find optimal SVM *j* weights per topic last year [9], but this was computationally unrealistic for our implementation. Particular choices of *n* we used for cross-validation are noted below.

Batch Using Linear SVMs with TF/IDF Vectors

For the submitted *apl11FbF* run, we used the top 20000 df terms. We used all the presumed negative training examples from the training index. We used three-fold cross-validation on the training data to select the best topic-specific score thresholds for the T11F measure.

For the submitted run *apl11FbSU* run, we used the top 12000 df terms. We again used all the presumed negative training examples from the training index. We used five-fold cross-validation on the training data to select the best topic-specific thresholds for the T11SU measure.

	T11SU	T11F	SetPrec	SetRecall
apl11FbF	0.391	0.216	0.409	0.117
apl11FbSU	0.293	0.181	0.244	0.255

Table 3. APL Batch Results, Assessor topics

	T11SU	T11F	SetPrec	SetRecall
apl11FbF	0.338	0.026	0.068	0.013
apl11FbSU	0.035	0.028	0.027	0.275

Table 4. APL Batch Results, Intersection topics

Adaptive Filtering Task

We developed two heuristic approaches to using SVMs for adaptive filtering. While there is theory to explain how a static SVM generalizes to test data of similar distribution to training data, this theory has not yet been well developed for SVMs that are adapting over time based on feedback.

Our two approaches were similar to early filtering score-based buffer window approaches. An SVM was created based on training data for each topic. Three "buffers" of documents from the test document stream were maintained: a "good" buffer of documents correctly judged relevant; a "bad" buffer of those that had been incorrectly judged relevant; and a "presumed bad" (unjudged) buffer of those documents not retrieved (and presumed irrelevant).

All buffers were capped to a fixed size. Window sizes for the buffers were set somewhat arbitrarily based on limited experimentation as follows: 750 documents for the known positive documents, 750 for the known negative documents and 2000 or 50 (noted below) for the presumed negative documents (those not retrieved). We also used a heuristic parameter to guess occasionally if no documents had been retrieved for a long time.

Our first approach used queues for all three of these buffers of documents, expiring the old documents as the buffers overfilled. The notion here is that older documents are less valuable than newer ones. (In this case we used the larger size 2000 buffer for negative documents.) Our second approach used heaps, based on the absolute value of the SVM score (smallest value on top), throwing away documents with larger scores as the buffers overfilled. The notion here is that documents closer to the margin of the current SVM are more useful as discriminative examples for training. (Here we used the smaller buffer size of 50 for the presumed negative documents.)

Our strategy was to update the topic-specific SVMs at data-driven intervals, using the documents in the current buffers. The intervals were based on sizes of the current buffers, as well as a 'rate of change" heuristic.

Using the queue approach we had observed good (but statistically variable performance) based on the TREC 2001 data and topics (0.35 average T10SU, and a boxplot as good or better than the official boxplots from TREC-10 filtering). However, we did not do as well for this year's official adaptive task.

	T11SU	T11F	SetPrec	SetRecall
apl11Fah1	0.342	0.104	0.377	0.039
apl11Fah2	0.342	0.104	0.377	0.039
apl11Faq1	0.059	0.09	0.084	0.369
apl11Faq2	0.085	0.118	0.115	0.355

Table 5. APL Adaptive Results, Assessor topics

Clearly, the heap approach returned too few documents, whereas the queue approach returned too many. This is probably mainly due to the much lower amount of feedback. It was probably also adversely affected by our choice of "guess occasionally" parameter that guessed too often.

Filtering Results Discussion

In a low training/feedback situation, filtering seems to require more of a Statistical Language Model score-based approach. Based on the good performance possible in situations with lots of training and feedback (as in TREC-2001), there seems to be a continuum between score-based and classification approaches, depending on the amount of training and feedback available. We conjecture a hybrid approach will be useful to support this continuum.

Arabic Language Retrieval

The Cross-Language Retrieval task at TREC 2002 consisted of bilingual retrieval of Arabic newspaper articles given English topic statements. The document collection was the same as that used in the TREC 2001 CLIR Track. Monolingual submissions were also accepted using Arabic versions of the topics created by human translators. JHU/APL submitted five official runs; one monolingual and four bilingual runs that used only the <title> and <desc> topic fields. We continued to use the HAIRCUT retrieval engine for our experiments, again emphasizing language-neutral approaches to multilingual retrieval.

Tokenization

Over the past year several studies explored alternate representations for indexing Arabic text. Mayfield *et al.* [10] investigated the use of character n-grams for Arabic retrieval in TREC-2001 and found that n-grams of length 4 were most effective. Similarly, Darwish and Oard examined multiple tokenization strategies for retrieval of scanned Arabic documents and concluded that character n-grams of lengths 3 or 4 were the basis for the most successful approach [1]. Linguistic methods of combating Arabic morphology have also been fruitful. Xu et al. [14] investigated several problems unique to Arabic language text retrieval, specifically misspelled words, broken plurals, and infix morphology, and empirically evaluated techniques to overcome them. Larkey et al. [8] investigated methods for effectively stemming Arabic.

Given the successful reports of n-gram based retrieval for Arabic, we opted to continue using them this year. However, we decided to use a combination of tokenization methods in the same term space. We used n-grams of more than one length, and we included space-delimited words. We do perform one minor language-specific function, elimination or replacement of certain Arabic characters. Specifically, we map Alef Maksura to Yeh and Teh Marbuta to Teh, and we eliminate Hamza, Madda and any remaining Arabic letters or symbols that did not appear in a list of 28 letters that we had available.

Recent work in Asian language retrieval has shown that multiple length n-grams can be quite effective, and may result in a 10% relative improvement in mean average precision over the use of single length n-grams [12]. Accordingly, we examined multiple length n-grams. In particular, we construct the set of all 3-grams, 4-grams, and 5-grams that can be generated from a given input sequence.

We initially built several indexes to compare different methods for tokenization. Summary information about each is shown in Table 6.

	# terms	index size
words	539979	254MB
3-grams	27016	441MB
4-grams	225218	766MB
5-grams	1478593	1157MB
6-grams	6081618	1691MB
words + 3/4/5-grams	2876187	2422MB
words + 3/4/5/6-grams	9714673	4038MB

Table 6. Index statistics for the 869 MB, 384K article TREC-2002 Arabic collection.

Using the TREC-2001 CLIR test collection (*i.e.*, Arabic topics 1-25) we compared several knowledge-light methods for indexing Arabic text (see the chart in Figure 1). These experiments used only the <title> and <desc> portions of the topic statements and made use of pseudo-relevance feedback. Plain 4-grams did quite well, but slightly superior performance was found when a hybrid indexing scheme was used. Based on these training experiments, we selected this strategy for TREC-2002.

Thus, our official runs used both words and 3-, 4-, 5-grams to represent text in a single term-space. It should be noted that this tripled the disk space consumed by the index data structures compared to the use of solitary 4-grams; the use of 4-grams alone is probably justified when storage limitations are a concern.

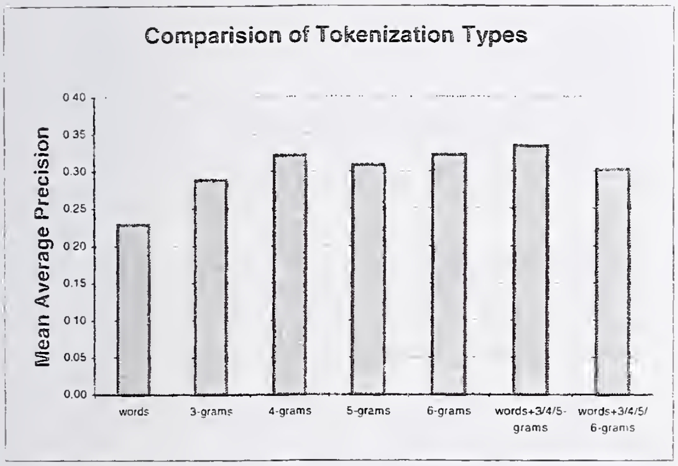


Figure 1. Comparison of tokenization methods using the TREC-2001 CLIR test suite. Mean average precision is plotted. The combination of words plus 3-, 4-, and 5-grams was the best performing approach.

Translation

Although we recently explored efficient methods for translating document representations at the CLEF-2002 evaluation [11], we focused on query translation for our work in the CLIR Track at TREC. We are convinced that the caliber of translation resources has a great effect on bilingual retrieval performance, so we were glad to see the track guidelines stipulate a standard set of resources. However, in several ways the formatting of these resources prevented us from using them in an optimal fashion. In particular, we had hoped to use the English / Arabic parallel texts from the United Nations. We were grateful for the statistical lexicon that was made available by BNN; however, it was of limited use to our system since we do not routinely stem English or Arabic.

Most of our bilingual runs simply relied on machine translation software. However, in an attempt to make use of the BBN statistical lexicon, we derived a surrogate dictionary. We first ran a Porter stemmer to create a set *E* of English words that could produce a given English stem; we also created a set of Arabic words *A*, that created the stems in BBN's lexicon using Kareem Darwish's Al-Stem stemmer. Then, we created an unweighted translation dictionary with entries between each English word in *E* and every word in *A* to which that word *might* be mapped. Queries were translated by substituting all possible translations for a given source language query term, preserving the original query term frequency. Lastly, we performed n-gram processing over the translated queries using only within-word n-grams.

Each of our official submissions used only the <title> and <desc> fields, augmented by pseudo-relevance feedback. For our monolingual Arabic run, *apl11ca1*, we used

- word plus 3-, 4-, and 5-gram indexing
- relevance feedback using queries expanded to 300 terms

Apl11ce1 was our first bilingual run using the English topics. We used the same approach as *apl11ca1*, but used the Almisbar web-based service to create translated queries. We also created a run using the (standard) Ajeeb translator, *apl11ce3*. Mappings derived from the statistical lexicon provided by BBN were used for *apl11ce4*. Finally, hoping that a combination of resources would maximize lexical coverage, and thus retrieval performance, we submitted a run based on merging scores from our two MT-based runs, *apl11ce2*. This run was not our best official run; use of only the standard MT-resource, the Ajeeb translator, was best.

Official results

An overview of APL's five official runs for the Arabic track are shown in Table 7 below.

	Trans Res.	MAP	Recall (5909)	# best	# ≥ median	% mono
apl11ca1	NA	0.3410	4977	3	29	100.0
apl11ce1	Almisbar	0.2427	4396	0	20	71.2
apl11ce2	Almisbar & Ajeeb	0.2571	4488	2	18	75.4
apl11ce3	Ajeeb	0.2658	4444	0	21	77.9
apl11cf1	Stat. Lexicon	0.1777	3645	1	11	52.1

Table 7. Official results for Arabic runs (50 topics). The highlighted rows indicate bilingual runs that used only standard translation resources.

Figure 2 (below) compares our monolingual run against the median of 18 monolingual runs.

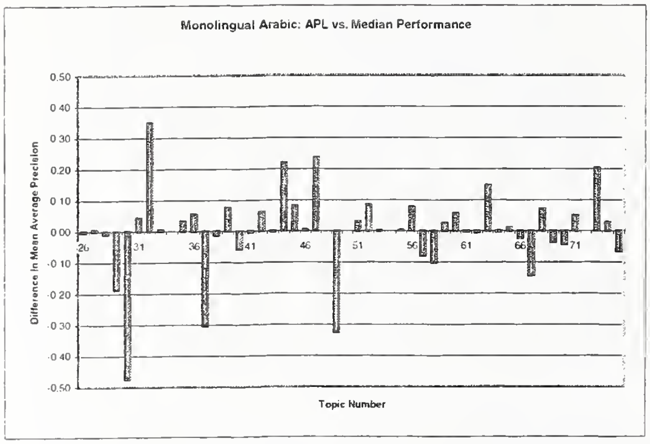


Figure 2. Comb chart for *apl11ca1*

The MT-based runs obtained performance between 71% and 78% of a monolingual baseline in terms of mean average precision; a relative recall at 1000 documents of 88% was found. A precision-recall graph comparing these results is plotted in Figure 3.

Conclusions

This year we participated in two tracks: filtering, and Arabic.

We continued our investigation of using Support Vector Machines (SVMs) to tackle text filtering challenges. We found promise for the use of SVMs for relevance feedback for routing. We plan to further investigate related SVM pseudo-relevance feedback effects on ad hoc retrieval. Our batch results appeared to be about median, and we had many "zero returns," so more remains to be done to tune this approach for low training situations. Perhaps Financial Times data was not similar enough to the evaluation data for use in parameter selection.

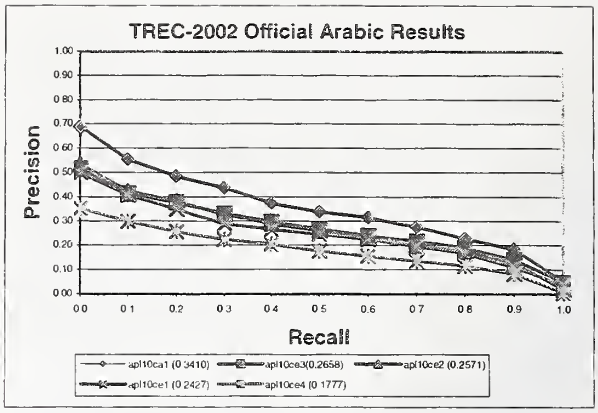


Figure 3. Recall-precision graph for APL's official Arabic track automatic submissions

Our adaptive filtering results were disappointing compared to what we had observed on TREC 2001 adaptive topics, although somewhat expected based on Financial Times parameter-setting experiments. Again, this is related to the much smaller amount of feedback in the track this year. It is possible to make better use of unlabeled (unjudged) data for SVM training, and we hope to revisit this in future experiments.

One thing we have observed in our CLIR work is that it is difficult to define standard translation resources. For example, it has proved difficult this year to separate specific stemming algorithms (and implementations) from some of the standard resources. We also wonder whether cross-system comparisons would be facilitated if participants submitted runs that used only a single translation resource. For the TREC-2001 CLIR guidelines, systems could use any of the three options (dictionary, statistical lexicon, or MT system), thus giving 7 ways to use 'standard' resources.

References

[1] K. Darwish and D. W. Oard, 'Term Selection for Searching Printed Arabic'. In the Proceedings of the25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-2002), Tampere, Finland, pp. 261-268, 2002.

[2] S. Dumais, J. Platt, D. Heckerman, M. Sahami, 'Inductive Learning Algorithms and Representations for Text Categorization," in Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM 98) (1998).

[3] Marti A. Hearst. Trends and controversies: Support vector machines. IEEE Intelligent Systems, 13(4):18-28, 1998.

[4] D. Hiemstra and A. de Vries, 'Relating the new language models of information retrieval to the traditional retrieval models.' CTIT Technical Report TR-CTIT-00-09, May 2000.

[5] T. Joachims, Making large-Scale SVM Learning Practical Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT Press, 1999.

[6] T. Joachims, "Text categorization with support vector machines: learning with many relevant features," in *Proc. 10th European Conference on Machine Learning ECML-98* (1998).

[7] W. Kraaij, 'TNO at CLEF-2001'. In Results of the CLEF-2001 Cross-Language System Evaluation Campaign (Working Notes), Darmstadt, Germany, pp. 29-40, 2001.

[8] L. S. Larkey, L. Ballesteros, and M. E. Connell, 'Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-Occurance Analysis'. In the Proceedings of the25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-2002), Tampere, Finland, pp. 275-282, 2002.

[9] Dave Lewis, personal communication, TREC 2001 Batch Filtering Task Experiments.

[10]James Mayfield, Paul McNamee, Cash Costello, Christine Piatko, and Amit Banerjee, JHU/APL at TREC 2001: Experiments in Filtering and in Arabic, Video, and Web Retrieval. In E. Voorhees and D. Harman (eds.), *Proceedings of the Tenth Text REtrieval Conference (TREC 2001),* Gaithersburg, Maryland, July 2002.

[11]Paul McNamee and James Mayfield, 'Scalable Multilingual Information Access', Draft version in the *Proceedings of the CLEF-2002 Workshop*, Rome, Italy, 19-20 September 2002

[12]P. McNamee, 'Knowledge-light Asian Language Text Retrieval at the NTCIR-3 Workshop," *Working Notes of the 3rd NTCIR Workshop*, 2002.

[13]D. R. H. Miller, T. Leek, and R. M. Schwartz, 'A Hidden Markov Model Information Retrieval System.' In the Proceedings of the 22nd International Conference on Research and Development in Information Retrieval (SIGIR-99), pp. 214-221, August 1999.

[14]J. Xu, A. Fraser, and R. Weischedel, 'Emprical Studies in Strategies for Arabic Retrieval'. In the Proceedings of the25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-2002), Tampere, Finland, pp. 269-274, 2002.

[15] http://ais.gmd.de/~thorsten/svm_light/

[16] http:/english.ajeeb.com/

[17] http://www.almisbar.com/salam_trans.html

[18] Reuters Corpus, volume 1: English Language, 1996-08-20 to 1997-08-19. We gratefully acknowledge the provision of the research corpus by Reuters Limited; without it, our filtering experiments would not have been possible.

Finding Named Pages via Frequent Anchor Descriptions

J. Malawong **A. Rungsawang**

Massive Information & Knowledge Engineering
Department of Computer Engineering
Faculty of Engineering
Kasetsart University, Bangkok, Thailand

Email: {g4565043, fenganr}@ku.ac.th

Abstract

This article describes about finding documents of interest via frequent anchor descriptions that being derived from the ".gov" web collection. The main idea of our approach is that we consider frequent anchor descriptions as documents. To find out the frequent item sets, we apply the Apriori algorithm with a new scoring criterion, called the *maximum correspondence*. We likewise integrate both retrieval scores calculated from anchor descriptions and title texts of the web pages to identify the resulting named pages, and foundthat these combination scores can boost the precision performance. Concluded from our preliminary experiments , this approach yields a considerable efficiency of named page finding in the aspect that it also highly reduces the document search space.

1 Introduction

The information searching generally focuses on the quantity of information that the user obtains. On the other hand, named page finding only focuses on the most significant web page represented as the answer to a query. However, named page finding needs higher precision performance. Therefore, finding a named page is a more difficult task than other on-line document retrieval. Named pages are frequently named by anchor descriptions and those pages are usually linked to with a great number of web pages. The consequence is that those pages obtains a variety of anchor descriptions. The appropriate name of each page is able to be assumed to be the name with the most frequent occurrence. With the according reasons, document representation, anchor descriptions, or title texts, for instance, is noticeably nifty for addressing the relevant named pages.

In this paper, we propose a novel approach to enhance the retrieval performance of named page finding through using frequent item sets. We consider the anchor descriptions and title texts as the item sets. Each item set represents the name of the document. Afterward, we discover the frequent item sets using the Apriori [6]. We then apply the new scoring method, i.e. the *maximum correspondence* function, to address the most significant frequent item set. In addition, we also study from resulting experiments on the integration of anchor descriptions and title texts to enhance the named page retrieval precision.

We organize this article in the following way—Section 2 describes about the document representations. Section 3 describes about the frequent item set discovery algorithm. Section 4 is dedicated for more details we supplement to the algorithm. Section 5 describes about the obtained experiment result. Section 6 finally concludes the paper.

2 Document Representations

There are many categories of document representations; for example, contents, abstracts, title texts, anchor descriptions, links, the depth of URL addresses, and the integration of those [1, 2], each of which has a potential of improving the retrieval performance. Nonetheless, there are advantages and disadvantages of each. In this work, we select the anchor descriptions and the title texts. It is due to the fact that they offer tremendous improvement of retrieval performance with the support of relatively small size of document representation.

2.1 Anchor Descriptions

A *hyperlink* is a relationship between two documents or two fragments of the same document. In the hypertext markup language (HTML), the target is referred by the link and the link's anchors are displayed to the users with the underlined text. As soon as the user click the anchor, their browser will display the target document. We call a link anchor appearing on the browser as an *anchor description.* The anchor description always describes its target. Moreover, a target is likely to be associated by other web documents and initiates considerable anchor descriptions if its target is popular.

Anchors	Occurrences
Kasetsart University	1,500
Kasetsart	834
Kasetsart Home Page	322
KU	301
KU City	115
Main	75
Home	24

Table 1: An example of anchor descriptions.

Previous studies [3, 4] found that anchor descriptions has a potential to enhance the retrieval performance. Therefore, we are in agreement to construct anchor documents [3] as a representative of each document in data collection for a substitution of the term. Each anchor document contains all anchor descriptions of a page's incoming links. Table 1 illustrates an example of the anchor document of the web page `www.ku.ac.th`. In this paper, we will call each entry as an *anchor description set.* Moreover, we will represent it with the notation $N \times \Lambda$, where N is the number of occurrences and Λ is the element sequence of the anchor descriptions. For example, the first entry of Table 1 can be represented with the notation 1500 × Kasetsart University. It is noticeable that each anchor description is the page's name appearing on the browser. The anchor description sets consequently represent the associativity between the link's anchor and its name.

2.2 Title Texts

The titles of web pages can be used as the document representation and it yields comparable, or even superior, retrieval results than full-text retrieval. Previous study [5] extracted several tag fields from data collection and found that the `<title>` field could produce better performance comparing with full-text retrieval. Furthermore, the retrieval system takes less substantial time and space. Therefore, we extract the title fields from the data collection and amass them similarly to the anchor description sets. We nevertheless separate the extracted title texts from those.

2.3 Size of Document Representations

We provide the quantitative measurement of the full text, title texts, and anchor descriptions derived from the .gov collection in Table 2. It is noticeable that anchor descriptions and title texts are relatively smaller than the whole collection. Moreover, they also contain relationship among links and their names regardless of document contents. This facilitates searching relevant named pages in collection containing only documents' names. As a result, we will likely obtain higher recall performance.

3 Frequent Item Set Discovery

Discovery of frequent item sets is the principle spirit of a great number of data mining approaches and it has been well studied in the context of association rules [7]. We usually decompose the problem solution of association rule mining into two phases—discovery of frequent item sets and rule generation from the discovered frequent item sets.

3.1 Frequent Item Sets

Let $I = \{i_1, i_2, i_3, \ldots, i_m\}$ be set of literals, so-called *items*. Let a non-empty set of items T be called an *item set* or a *transaction*. Let database D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. Each transaction has its statistical significance, so-called *support value*. The support of the item set X in the database D is defined in Equation 1.

$$\text{support}(X, D) = \frac{|\{T \in D | X \subseteq T\}|}{|D|} \qquad (1)$$

Let X be an item set. A transaction T is said to contain X if and only if $X \subseteq T$. The support of an item set X is the number of the percentage of transaction in the database that contains X. X is frequent if the support of X is not less than a user-defined support threshold, called *minimum support*.

Example Let Table 3 be a transaction database D, with a set of items $I = \{a, b, c, d, e, f, g, h\}$. Let the minimum support be 20 percents; the consequent frequent item sets are listed in Table 4.

3.2 Apriori Algorithm

The algorithm called *Apriori* [6] iteratively generates all possible item sets whose supports qualify the minimum support threshold. The first iteration of the algorithm counts item occurrences in order to generate the frequent 1-item sets (each 1-item set exactly contains only one item). For each next iteration, the frequent item set L_{k-1} found in the $(k-1)$th iteration are used to generate the candidate item sets C_k, using `apriori-gen` function described in Algorithm 1.

Representation	Size	Nb. Doc	Avg. Len.
Full Text	19,455 MB	1,247,753	15.2 KB
Title Text	148 MB	893,544	0.17 KB
Anchor Description	3,302 MB	827,256	4.0 KB

Table 2: Quantitative measurement of each document representation derive from the .gov collection.

Trans. ID	Item sets
1	a, b, c, d, f
2	b, c, d, f, g, h
3	a, c, d, e, f
4	c, e, f, g

Table 3: An example of a Transaction database D

Length (l)	Frequent l-item sets
1	a, b, c, d, e, f, g
2	$ac, ad, af, bc, bd, bf, cd, ce, cf, cg, df, ef, gf$
3	$acd, acf, adf, bcd, bcf, cdf, cef, cfg$
4	$acdf, bcdf$

Table 4: Frequent item sets with 20-percent minimum support derived from transaction database D.

Afterward, the database is scanned and the support of candidates in C_k is counted. The output of the first phase of Apriori algorithm consists of a set of k-item sets (where $k = 1, 2, 3, \ldots$), whose supports qualify the specified minimum support threshold. Algorithm 1 presents a formal description of the algorithm. We assume that items in each item set are lexicographically sorted.

Algorithm 1 The Apriori Algorithm

```
Scan $D$ to find $L_1$.
Let $k = 2$.
while $L_{k-1} \neq \emptyset$ do
    $C_k$ = apriori-gen($L_{k-1}$).
    for all transaction $t \in D$ do
        $C_t$ = subset($C_k$, $t$).
        for all candidate $c \in C_t$ do
            $c$.count = $c$.count + 1.
        end for
    end for
    $L_k = \{c \in C_k | c.\text{count} \geq \text{minsup}\}$.
end while
return $\bigcup_k L_k$.
```

Candidate item sets C_k are generated from previously generated frequent item sets L_{k-1} through using the apriori-gen function. The apriori-gen performs two operations, as follows.

Joining step: Each large item set from L_{k-1} is joined together.

Pruning step: Each item set $c \in C_k$ such that some $(k-1)$-subset of c, in which L_{k-1} does not contain, is deleted.

4 Contributions in Frequent Item Set Discovery

In this article, we will focus on the TREC-11 named page finding mission. We separate the method into three phases, as follows.

4.1 Document Representation Extraction

In this phase, we extracted the document representation, i.e. anchor descriptions and title texts, from TREC-11 data collection suite. Current data collection (Year 2002) is a crawler's outcome concentrated on the government domain (.gov websites) and it consists of 1.25 million documents. We analyzed links and their anchors from each document. Afterward, we kept those in their targets' anchor document. We then constructed an anchor description collection with every document. An anchor document collection comprises of 5.5 million documents. It contains anchor documents in data collection and contains anchor documents created from links whose target does not occupy in the data collection. Moreover, we filtered out the anchor documents whom the data collection does not contain, as well. After that, we subsequently performed morphological analysis process. We filtered out the stop words and every special sign except the hyphen (-), but we did not, in contrast, perform lexicon stemming. Finally, an anchor document collection contains only 0.8 million documents. We also extracted the title texts from the data collection. We kept them in the collection analogous to the anchor documents.

4.2 Frequent Anchor Description Discovery

Let $I = \{w_1, w_2, w_3, \ldots, w_n\}$ be a set of words in an anchor document. Let T be an item set that $T \subseteq I$. For example, the set of anchor descriptions representing Kasetsart University web site is shown in Table 5.

Let the minimum support be 20 percents. We discovered the frequent anchor descriptions via the Apriori algorithm (see Algorithm 1). We provide the consequence of applying the Apriori to data in Table 5 in the Table 6. We then extract the frequent anchor descriptions from anchor documents in order to provide an availability of retrieval. For title text collection, in view of the fact that each document possesses only one item set, we will permanently judge it to be frequent.

4.3 Relevant Named Page Retrieval

For the reason that the present retrieval methodology exploiting term frequencies and their weights in the data collection does not have a capability to rank the frequent anchor descriptions for this system, we therefore develop the *maximum correspondence* function, as in Equation 2.

$$\Psi_i = \max_j \vec{q} \cdot \vec{\lambda}_{ij} \tag{2}$$

Where, Ψ_i is a maximum correspondence among the query $\vec{q}$ and each $\vec{\lambda}_{ij}$ anchor description of the document i.

The equation ranks each document with correspondence between the query with each frequent anchor description. The maximum score from every frequent anchor description is the score of the document. We likewise apply the Equation 2 with the title text collection.

5 Experimental Results

With the intention of having a manageable task, we decomposed the evaluation phase into three experiments. We evaluated the system with only the frequent anchor description approach, then the title texts, and finally the integration of both. 150 queries are automatically prepared from TREC-11 named page finding task's queries without stems. We illustrated the retrieval result in Table 7 and Figure 1. The distribution of the result is depicted in Figure 2.

From Table 7, the frequent anchor description outperforms the title texts collection, since the frequent anchor description has a capability to provide more documents' names considered as exact names. Nonetheless, some documents do not have or have inadequate anchor descriptions due to incomplete links. These problems essentially affects the retrieval performance. On the other hand, we found that web documents always have title texts that can likewise perform well in case of the mentioned problems. Therefore, it is noticeable that the integration of both yields the better performance, as shown in Figure 1 and Figure 2. The ranks of relevant named pages are boosted to the top rank and more relevant named pages can be found.

6 Conclusion and Future Works

The frequent anchor descriptions can represent the document's name with relatively small size of data. However, the incomplete link problem principally affects the reduction of performance. The title texts can elucidate this problem, but it provides inadequate information, in contrast. The integration of both can provide better exper-

Item Set ID	Anchor Description
1	Kasetsart University Page
2	Homepage of Kasetsart University
3	Kasetsart The University of Agriculture
4	Kasetsart University Page
5	Homepage of Kasetsart University
6	Homepage of Kasetsart University
7	Kasetsart University Page

Table 5: Anchor description representing Kasetsart University's web site

Length (l)	Frequent l-anchor description
1	{ Kasetsart }, { University }
2	{ Kasetsart University }, { Kasetsart of }, { Kasetsart page }, { Kasetsart Homepage }, { University Page }, { University Homepage }, { University of }
3	{ Kasetsart University of }, { Kasetsart University Page }, { Kasetsart University Homepage }, { Kasetsart Homepage of }. { University Homepage of }
4	{ Kasetsart University Homepage of }

Table 6: Frequent anchor descriptions of Kasetsart University's web site

Ranks	Titles	Anchors	Integration
Top 1	20	56	64
Top 5	24	19	20
Top 10	9	8	10
Top 25	13	9	13
Top 50	5	4	4
Not Found	79	54	39
MRR	0.406	0.588	0.680

Table 7: Named page finding retrieval result

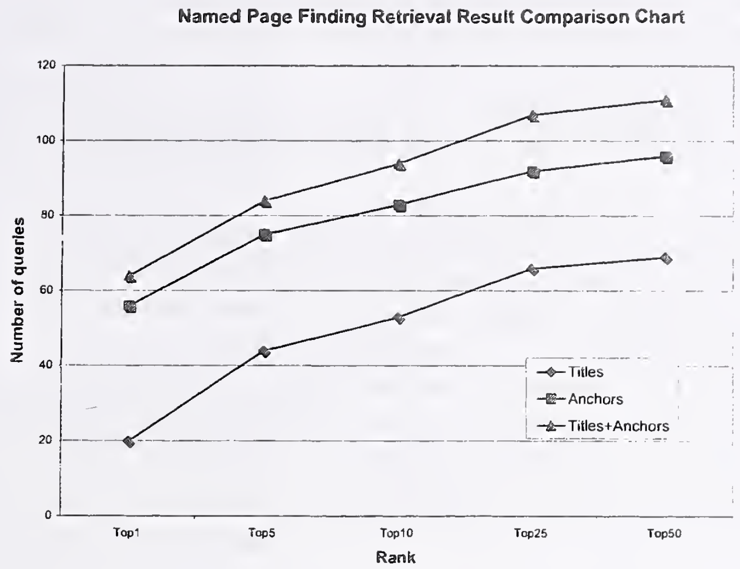


Figure 1: Named page finding retrieval result comparison chart

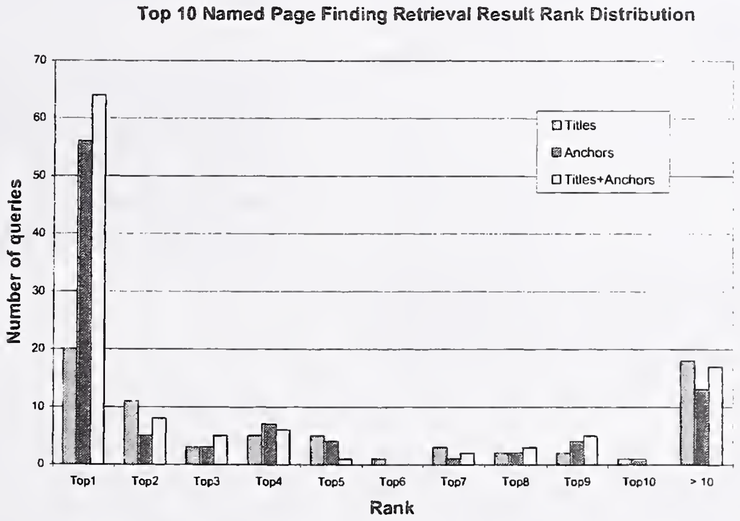


Figure 2: Name paged finding retrieval result distribution

imental result due to the management ability of incomplete links and relatively small size of data comparing with full text data. We concluded that the frequent anchor description provides a valuable source of information for named page finding task.

The frequent anchor description can enhance the retrieval performance for commercial search engines, but the maximum correspondence function operates very lingeringly on simple frequent anchor description file structure. We consider this our future work. The association rule discovery from frequent anchor description can provide confidence value for the scrutiny of strong frequent anchor description. Moreover, it can reduce the size of frequent anchor description collection.

Acknowledgement

We would like to thank all MIKE staffs for their comments and working spirit. We are much obligned to Prachya Boonkwan for his kindness in reviewing this paper.

References

[1] J.A. Shaw and E.A. Fox, *Combination of multiple searches*, in Proceedings of the 3rd Text REtrieval Conference (TREC-3), pp. 105-115. Gaitherburg, MD: National Institute of standards and Technology, 1995.

[2] C.C. Vogt and G.W. Cottrell, *Predicting the performance of linearly combined IR systems*, in Proceedings of the 21st annual international ACM SIGIR conference on Research and Development in Information Retrieval, pp. 190-196, New York: ACM, 1998.

[3] N. Craswell, D. Hawking and S. Robertson, *Effective Site Finding using Link Anchor Information*, in Proceedings of 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 250-257, 2001.

[4] Sergey Brin and Lawrence Page, *The anatomy of a Large-scale hypertextual web search engine*, in Proceedings of WWW7, 1998.

[5] W. Xi and E.A. Fox, *Machine Learning Approach for Homepage Finding Task*, in Proceedings of the 10th Text REtrieval Conference (TREC-10), pp. 686-697, Gaithersburg, MD: National Institute of standards and Technology, 2001.

[6] R. Agrawal and R. Srikant, *Fast algorithms for mining association rules*, in Proceedings of VLDB'94, pp. 487-499, Santiago, Chile, 1994.

[7] R. Agrawal, T. Imielinski and A. Swami, *Mining Association Rules Between Sets if Items in Large Database*, in Proceeding of ACM SIGMOD International Conference on Management of Data, pp. 207-216, Washington DC, USA, 1993.

Kernel Methods for Document Filtering

Nicola Cancedda[‡], Nicolò Cesa-Bianchi[*], Alex Conconi[*], Claudio Gentile[§],
Cyril Goutte[‡], Thore Graepel[†], Yaoyong Li[†], Jean-Michel Renders[‡], John Shawe-Taylor[†],
Alexei Vinokourov[†]

[§]CRII
Università dell'Insubria, Italy
<FamilyName>@dsi.unimi.it

[†]Department of Computer Science
Royal Holloway, University of London
<FirstName>@cs.rhul.ac.uk

[*]Dipartimento di Tecnologie dell'informazione
Università degli Studi di Milano
<FamilyName>@dti.unimi.it

[‡]Xerox Research Centre Europe
<FirstName>.<FamilyName>@xrce.xerox.com

4th February 2003

Abstract

This paper describes the algorithms implemented by the KerMIT consortium for its participation in the TREC 2002 Filtering track. The consortium submitted runs for the routing task using a linear SVM, for the batch task using the same SVM in combination with an innovative threshold-selection mechanism, and for the adaptive task using both a *second-order perceptron* and a combination of SVM and *perceptron with uneven margin*. Results seem to indicate that these algorithm performed relatively well on the extensive TREC benchmark.

1. Introduction

The KerMIT IST European project is concerned with the investigation of kernel methods for applications related to the categorization, retrieval, clustering and ranking of text documents and of images[1]. The KerMIT consortium participated in the TREC 2002 Filtering track as a means of evaluating the methods developed within the project on a large-scale benchmark. Six runs were submitted, out of which four for the adaptive task and one each for the batch and for the routing task. As the objective of our participation was the comparison of different techniques, submitted runs are actually issued from several different systems:

- Runs 'af1' and 'af2' were obtained using a variant of the "second-order perceptron" (Cesa-Bianchi et al. 2002)(Section 3);
- Runs 'af3' and 'af4' were obtained using a combination of the SVM algorithm and the Perceptron Algorithm with Uneven Margin (Li et al. 2002)(Section 4);
- Runs 'bf2' and 'rr2' are the output of SVMs, the former in combination with an improved threshold-selection mechanism (Section 5).

The paper is organised as follows. Section 2 sketches the data preparation process. Sections 3 to 5 describe the systems listed above in turn, together with the performance achieved on the respective tasks. Section 6 presents some additional experiments on intersection topics. Section 7 contains some concluding remarks.

2. Data preprocessing

Before turning to the description of the individual systems, we will outline the data preparation process

[1] More information on the KerMIT IST project is available from the project website:
http://www.euro-kermit.org.

that we followed. The original NewsML files are preprocessed in the following way:

- The "title" and the "text" portions of the files are extracted and cleaned from tags;
- Each file is tokenised into words using a finite-state based tokeniser;
- All digit characters are replaced with a single special character;
- Stopwords are removed;
- A dictionary file is built associating a numeric code with each token occurring at least three times in the training set. Terms occurring only once or twice are ignored;
- For every document body and every title a sparse term vector is built;
- The title and the body vector for each document are combined giving double weight to the title;
- All document vectors are finally modified according to a tf*idf weighting scheme and normalised to unit norm.

In the case of the batch and the routing runs, idf is computed for every term based on the training set only. Idf weights for all other terms are set to zero. In other words, only terms in the test documents which also occur in the training set are considered.

In the case of the adaptive filtering runs, idf weights are initialized on the training corpus in the same way as for the batch and the routing runs. However, as more and more test documents come in, idf weights are updated. Similarly for the lexicon, new lexical items are added to the dictionary as soon as the number of occurrences in the training set combined with the test set up to the document itself reaches a threshold of three.

The adopted tf*idf weighting is the usual log-log one:

$$w_{i,j} = (1 + \log(tf_{i,j})) \log\left(\frac{N}{df_i}\right)$$

where $tf_{i,j}$ is the number of occurrence of term i in document j, N is the total number of documents in the collection, and df_i is the number of documents containing the term i.

The very limited availability of positive examples called for ways to take advantage of the topic descriptions as well. We considered two alternative approaches, one consisting in building an additional positive training example from the descriptions and another consisting in building a vector from the description and then adding to each document, as a new feature, the value of its (cosine) similarity with the description. The second alternative proved superior and was thus retained for the batch and the routing tasks, whereas time constraints did not allow its adoption in the adaptive case.

3. The second-order perceptron algorithm for adaptive filtering

In this section we describe the first of the two algorithms used in the adaptive filtering track. This first algorithm is defined by a pair $(\boldsymbol{w}, \tau)$, where $\boldsymbol{w} \in \mathbb{R}^N$ is the profile vector and $\tau \in \mathbb{R}$ is the relevance threshold. A document $\boldsymbol{x} = (x_1, \ldots, x_N) \in \mathbb{R}^N$ is judged relevant if and only if the *margin* $\boldsymbol{w}^\top \boldsymbol{x}$ (i.e., the inner product between $\boldsymbol{w}$ and $\boldsymbol{x}$) is not smaller than the threshold τ.

We model the filtering problem under the assumption that relevance judgments are generated using an unknown probabilistic linear function. Assuming all documents $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ are normalized such that $||\boldsymbol{x}_t|| = 1$ for all $t \geq 1$, the relevance of $\boldsymbol{x}_t$ is given by a $\{-1, 1\}$-valued random variable Y_t (where $Y_t = 1$ means "relevant") such that there exists a fixed and unknown "target" profile vector $\boldsymbol{u} \in \mathbb{R}^N$, $||\boldsymbol{u}|| = 1$, for which $\mathbb{E}Y_t = \boldsymbol{u}^\top \boldsymbol{x}_t$ for all $t = 1, 2, \ldots, n$, where $\mathbb{E}$ indicates the expected value. Hence $\boldsymbol{x}_t$ is relevant with probability $(1 + \boldsymbol{u}^\top \boldsymbol{x}_t)/2 \in [0, 1]$. The random variables $Y_1, Y_2, \ldots$ are assumed to be independent, whereas we do not make any assumption on the way the sequence $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ of documents is generated.

The profile vector of our filtering rule is a (biased) estimator of the target profile $\boldsymbol{u}$ constructed as follows. Let S_t be the matrix whose columns are the forwarded documents after the first t time steps and let $\boldsymbol{Y}_t$ be the vector of corresponding observed relevance labels. Note that $\mathbb{E}\boldsymbol{Y}_t = S_t^\top \boldsymbol{u}$ holds. Drop the index t for clarity and consider the least squares estimator $(S\,S^\top)^\dagger S\,\boldsymbol{Y}$ of $\boldsymbol{u}$, where $(S\,S^\top)^\dagger$ is the pseudo-inverse of $S\,S^\top$. For all $\boldsymbol{u}$ belonging to the column space of S, this is an unbiased estimator of $\boldsymbol{u}$, that is

$$\mathbb{E}\left[(S\,S^\top)^\dagger S\,\boldsymbol{Y}\right] = (S\,S^\top)^\dagger S\,\mathbb{E}\boldsymbol{Y} = (S\,S^\top)^\dagger S\,S^\top \boldsymbol{u} = \boldsymbol{u}\ .$$

To remove the assumption on $\boldsymbol{u}$, we make $S\,S^\top$ full rank by adding the identity matrix I. This also allows us to replace the pseudo-inverse with the standard inverse, obtaining the biased estimator

$$(I + SS^\top)^{-1}SY \tag{3.1}$$

with expectation $\mathbb{E}\left[(I + SS^\top)^{-1}SY\right] = u - (I + SS^\top)^{-1}u$ (this immediately follows from the matrix identity $(I + SS^\top)^{-1} SS^\top = I - (I + SS^\top)^{-1}$). Estimator (3.1) is a "sparse" variant of the ridge regression estimator (Hoerl and Kennard 1970), where the sparseness is due to the fact that we only store in S the documents for which we have a relevance label (i.e., those that were forwarded).

We use a variant of (3.1) that tries to estimate directly the margin $u^\top x$ rather than estimating u. More precisely, we estimate $u^\top x$ with the quantity $W^\top x$, where the profile vector W is defined by

$$W = (I + SS^\top + xx^\top)^{-1}SY \ . \tag{3.2}$$

Using the Sherman-Morrison formula, we can then write out the expectation of $W^\top x$ as

$$\mathbb{E}\left[W^\top x\right] = \frac{u^\top x - u^\top(I + SS^\top)^{-1}x}{1 + x^\top(I + SS^\top)^{-1}x}$$

which holds for all u, x, and all matrices S. Comparing the bias of W to the bias of (3.1) in estimating the margin, we may observe that W introduces a multiplicative bias whose effect is to shrink the expectation of the margin $W^\top x$. In fact, the term $x^\top(I+SS^\top)^{-1}x$ is always nonnegative due to the positive definiteness of $(I + SS^\top)^{-1}$. In the experiments W turns out to perform better than (3.1), though at present we do not have a convincing theoretical explanation of this fact. This algorithm can be turned into an equivalent dual form, which is needed when we use the feature expansion facility provided by the kernel functions. As a matter of fact, since the document vectors x in the dataset at hand have a large number of components, we found it convenient to run the dual form even without kernels. As a final remark, we note that W is strongly related to the second-order Perceptron algorithm for binary classification introduced in (Cesa-Bianchi et al. 2002).

We now move on to the choice of the threshold τ. A possible route, which has been followed in (Cesa-Bianchi et al. 2003), is to approximately compute for each document x an interval centered on $W^\top x$, around which $u^\top x$ falls with high confidence. Then, whenever x is such that the left-hand border of the interval for $W^\top x$ is negative, x is judged relevant and forwarded. This approach corresponds to setting τ to a negative value chosen as a function of both x and the current profile. The primary effect of this approach is to boost recall at the expense of precision, resulting in an increased net performance when precision and recall are scored the same (see the results in (Cesa-Bianchi et al. 2003)). However, this goes exactly against the TREC evaluation measures which put an emphasis on precision. To reduce recall we then decided to set τ to a positive (instead of negative) value in the interval $[0, 1/10]$. This choice reduces dramatically the number of forwarded documents, thus pushing precision up, but it also slows down the convergence of the profile to the target u, which results in a decrease of precision. Hence, unlike the one based on confidence intervals, this setting of τ, needs a reasonably good profile to start with.

3.1. TREC results

Based on the above discussion, we set the threshold τ to 0.1 for the first run (KerMITT11af1) and to 0.05 for the second run (KerMIT11af2). We then built an initial profile for each topic using a training set with 4 positive examples (three provided with the data plus one we built using the topic description). The table below shows the average results over TREC11 topics for KerMITT11af1 and KerMITT11af2 runs.

Topics range	KerMITT11af1		KerMITT11af2	
	T11U	T11F	T11U	T11F
Assessor	0.456	0.378	0.459	0.376
Intersection	0.323	0.049	0.310	0.047
All	0.389	0.213	0.385	0.211
Assessor ext. reljs	0.473	0.395	0.475	0.392

The last row contains results for assessor topics with the extended relevance judgments provided after the TREC conference. The positive threshold helped to control the number of false positives for the assessor topics (R101–R150), on which af1 and af2 obtained relatively good results. This did not happen on the intersection topics (R151–R200), where the average normalized linear utility measure (T11U) turns out to be slightly worse than the one of the trivial algorithm which retrieves nothing, and the $F_{0.5}$ measure (T11F) is remarkably low.

In the Reuters corpus with TREC11 topics only a small amount of positive examples is available for each topic. This leads to two problems:

- explorative predictions (i.e. when the algorithm judges a document as relevant because it is interested in obtaining the true label rather than scoring a good prediction) are not useful, because most of the times the obtained label is negative and conveys no information.

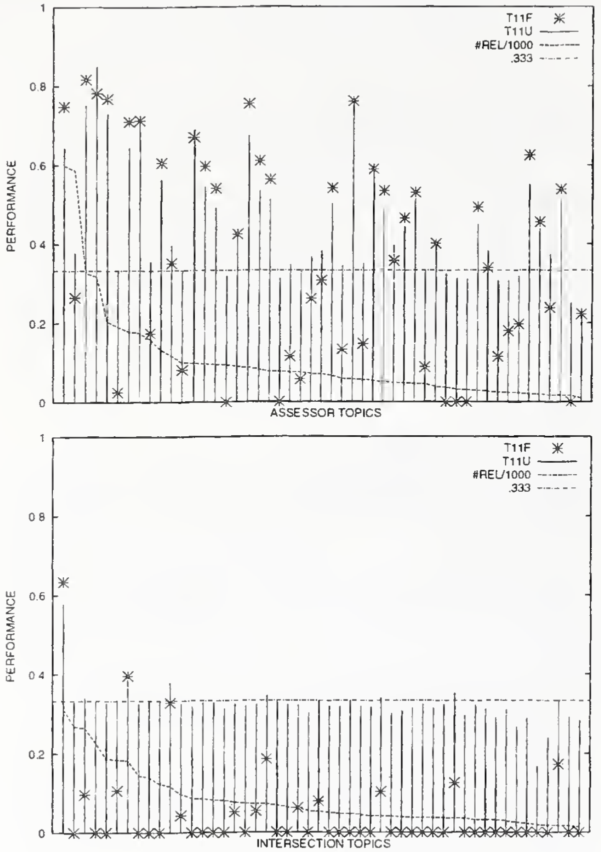


Figure 3.1. Performance for KerMITT11af1 on assessor (top) and intersection (bottom) topics.

- wrong predictions are very likely to be false positives, which badly affect the TREC evaluation measures.

Figure 3.1 plots the performance for the run KerMITT11af1 on each of the assessor and intersection topics. The topics (x-axis) are ordered according to their decreasing frequency. The horizontal line at .333 marks the T11U measure of the trivial algorithm which retrieves nothing. As the plots clearly show, the T11F measure exhibits a dramatic drop on almost all of the intersection topics (except a few of the most frequent ones). The T11U measure does not drop so badly on the intersection topics, even though its average value remains slightly below the .333 threshold.

Our future plan is to use a separate self-tuning threshold for each topic, using the current number of false positives as an indicator of whether it is better to bias the threshold towards positive or negative predictions.

4. SVM plus the Perceptron Algorithm with Uneven Margin for adaptive filtering

This section describes the second system implemented for the adaptive filtering task. In this system we adapted the Support Vector Machine (SVM) for the adaptive filtering. The SVM is quite suitable and successful in batch filtering (Lewis 2001), which is essentially normal text categorization (Joachims 1997). However, the adaptive filtering task in TREC is different from batch filtering in several aspects.

1. Only a small number of positive examples and a great number of unjudged documents are provided to create an initial profile [2].

2. The profile can be updated based on the retrieved documents for adaptive filtering, whereas it is fixed when it is applied to the test documents in batch filtering.

3. There are three kinds of documents in adaptive filtering, i.e. the positive, the negative and the unjudged, which can be employed to update the profile. The system can achieve good performance if it takes into account the different contributions of the documents toward the profile.

Our system deals with these issues by the following strategies.

1. The Gram-Schmidt algorithm (Cristianini et al. 2002) was employed to choose the negative examples from the unjudged training documents, i.e. these unjudged documents which are the furthest away from the given positive examples. This reduces the likelihood of choosing an example that is actually a positive example.

2. Use a fast and effective on-line version of the SVM, the Perceptron Algorithm with Uneven Margins (PAUM) (Li et al. 2002), to update the profile by the latest retrieved document.

3. Introduce several so-called margin parameters into the SVM as well as the PAUM to balance the contributions of different kinds of documents towards the profile.

[2] It is noted that the so-called one-class SVM can learn only from positive examples. However, the experiments have shown that the normal SVM using both the positive and negative examples can achieve much better performance in text categorization than the one-class SVM.

In detail, the algorithm used in our system is as follows.

- **Require:** n_g — the number of negative examples chosen for training.
  Require: γ_p and γ_n — the marginal parameters in the SVM for the positive and negative examples, respectively.
  Require: τ_p, τ_n and τ_u — the marginal parameters in the PAUM for the positive, the negative and the unjudged documents, respectively.
  Require: t — the threshold to retrieve the test document.
- **Training**
  - Training set — As we were given some positive training examples, we choose the negative training examples from the unlabeled training documents by the Gram-Schmidt algorithm. In detail, we apply the Gram-Schmidt algorithm to the given positive examples and compute the residual norms of the unjudged training documents. We then chose the first n_g documents with the largest residual norms as the negative examples.
  - Training method — After obtaining the training set, solve the corresponding SVM with uneven marginal parameters γ_p and γ_n:

    ```
    minimize_{w,ξ} $\langle \mathbf{w}, \mathbf{w} \rangle + C \sum_{i=1}^{l} \xi_i$
    subject to
    $\langle \mathbf{w}, \mathbf{x}_i \rangle + \xi_i \geq \gamma_p$ for the positive examples
    $\langle \mathbf{w}, \mathbf{x}_i \rangle - \xi_i \leq -\gamma_n$ for the negative examples
    $\xi_i \geq 0$      for $i = 1, ..., l$
    ```
- **Test**
  1. Apply the profile $\mathbf{w}$ to the test documents sequentially. For the test document $\mathbf{d}_i$, apply the profile $\mathbf{w}$ to the $\mathbf{d}_i$. If $\langle \mathbf{w}, \mathbf{d}_i \rangle > t$ then the document $\mathbf{d}_i$ is retrieved and is used to update the profile as shown in 2.
  2. Update the profile $\mathbf{w}$ using the marginal perceptron algorithm:

     ```
     Let $\tau = \tau_p$ and $y_i = +1$, if the $d_i$ is relevant.
     Let $\tau = \tau_n$ and $y_i = -1$, if the $d_i$ is irrelevant.
     Let $\tau = \tau_u$ and $y_i = -1$, if the $d_i$ is unjudged.
     while $y_i \langle \mathbf{w}, \mathbf{d}_i \rangle \leq \tau$
         $\mathbf{w} \leftarrow \mathbf{w} + y_i \mathbf{d}_i$
     endwhile
     ```

It is worth noting that, in the above algorithm, the profile is updated by using only the latest retrieved document d_i. We will discuss a variation of the algorithm later, which employs all the currently retrieved documents to update the profile.

4.1. TREC results

In order to actually apply the described algorithm to the TREC2002 dataset, the values of the parameters n_g, γ_p, γ_n, τ_p, τ_n, τ_u, and t need to be fixed. As part of previous experiments in the KerMIT project –before any actual work on the TREC2002 evaluation started– we had used part of the RCV1 corpus with the original reuters categories –as in the TREC2001 evaluation– to evaluate this same algorithm. More precisely, the training set consisted of the documents of the first 12 days, 20 through 31 August, 1996, and the test set was made of the documents of 30 days chosen from 1 October 1996 to 28 February, 1997. The topics considered in the dataset were those categories of the RCV1 classification scheme which had between 10 and 60 positive examples in the training set. Ten different parameter configurations were tried. The one which produced the best results was the following:

$$n_g = 80, \quad \gamma_p = 20, \quad \gamma_n = 5, \quad \tau_p = 10, \quad \tau_n = 2, \quad t = 5.0.$$

These values were retained for the KerMIT submissions based on this algorithm, namely KerMITT11af3 and KerMITT11af4, the assumption being that they were obtained through experiences in all comparable to a participation in the TREC2001 filtering track (see the TREC 2002 filtering track overview paper for a discussion of the issue of prior use of RCV1). It is indeed possible that results for those same runs would have been somewhat worse, had we used an entirely different document collection for chosing parameter values. Notice that a version of the algorithm without τ_u had been used in the previous experiments, given that the dataset we used did not contain any unjudged documents, and thus τ_u remained still to be assigned a value. It was decided to consider unjudged documents as "weekly negative" examples, and choose τ_u according to

$$\tau_u = -0.5 \frac{(pe_i + ne_i)}{(pe_i + ne_i + ue_i)},$$

where pe_i, ne_i and ue_i are the numbers of retrieved relevant, irrelevant and unjudged documents up to document i, respectively.

Finally, we adapted the threshold t in our system according to a simple rule whenever too few documents or too many unjudged documents were retrieved. The

run ID	Averaged over topics	T11SU	T11F	Precision	Recall
KerMITT11af3	Assessor topics	0.458	0.426	0.527	0.302
	Intersection topics	0.285	0.048	0.076	0.026
	Ass. top. (ext. reljs)	0.450	0.402	0.488	0.293
KerMITT11af4	Assessor topics	0.454	0.409	0.506	0.304
	Intersection topics	0.287	0.056	0.097	0.029
	Ass. top. (ext. reljs)	0.458	0.404	0.483	0.316

Table 1. The results of the two submitted runs. The last row for each run shows the result, on the assessor topics, with the extended relevance judgments provided after the TREC conference.

	Averaged over topics	T11SU	T11F	Precision	Recall
Run_1	Assessor topics	0.428	0.387	0.491	0.269
	Intersection topics	0.274	0.040	0.064	0.021
Run_2	Assessor topics	0.474	0.460	0.543	0.366
	Intersection topics	0.276	0.074	0.114	0.037

Table 2. The results of two additional runs, to be compared with the submitted run KerMITT11af3. Run_1 is obtained by choosing the negative training examples randomly. Run_2 is produced updating the profile by all the currently retrieved documents.

idea is that the system checks the threshold t everytime after a multiple of n_a test documents has been processed. In each check, if too few documents have been retrieved then the system decreases the threshold, otherwise, if too many unjudged documents have been retrieved, the system increases the threshold. We set $n_a = 10000$ in our system.

We applied the algorithm with the above settings to the dataset of TREC2002 adaptive filtering, and submitted two runs from this system, namely KerMITT11af3 and KerMITT11af4, with the initial values of the threshold t set to 5.0 and 4.5, respectively. The results for the two runs are listed in Table 1. We can see that the results are very different between the assessor topics (the first 50) and the intersection topics (the last 50).

4.2. Results of additional runs

In addition to the two submitted runs, we did several more experiments to test our algorithm. Table 2 lists the results of two additional runs.

One additional run was used to check the gain of the Gram-Schmidt algorithm in our system. This run had the same settings as those of KerMITT11af3, except that the negative examples for training were chosen randomly from the training documents exclusive of the three relevant documents. The results of this run, listed in the Run_1 of Table 2, show that selecting negative examples using the Gram-Schmidt algorithm yields significantly better results then selecting them at random.

In another additional run we obtained even better results than our two submitted runs from the system. In this run, we updated the profile by using all the currently retrieved documents, instead of only the latest retrieved document as we did in the two submitted runs. In detail, for each topic, when a new document was retrieved, we added it to the training set, and then apply the SVM with uneven margins to this updated training set to compute the new profile (i.e. the weight vector of SVM). The initial value of the threshold t in this run was set as 7.0, and other experimental settings are the same as those of KerMITT11af3. The result of this run is listed in Table 2, in the row marked as Run_2. It can be seen that, compared with the two submitted runs KerMITT11af3 and KerMITT11af4, we obtained better results in the Run_2 as we used more information to update the profile. On the other hand, Run_2 took much more processing time and needed more memory than the two submitted runs.

5. A new threshold-selection mechanism for the SVM for batch filtering

This section describes the method used for runs *bf2* and *rr2*, for the batch and the routing subtasks respectively.

Our choice for a basic classifier fell on the best scoring system of the TREC 2001 evaluation - the SVM^{light} package (Joachims 2002). We trained an SVM on the TREC preprocessed training corpus (see Section 2) and thus obtained predictions γ_i (distances from the separating hyperplane taken with the appropriate sign) of the labels for each document d_i from the training corpus. Note that training data were separable for all test queries and thus the sign of γ_i can be considered as the label.

The parameter C for SVM training was chosen based on machine learning theory (Vapnik 1998): $C_{opt} = ||R||^{-1}$ (where $R = \max ||x_i||$ is the radius of the ball containing training datapoints x_i and centered in the origin). As the data vectors were normalised C was set to 1.

Full cross-validation for determining the relative importance to be given to positive and negative examples in the training set (parameter j in SVM^{light}) was impractical given the size of the training set and the number of categories. We thus decided to use a fixed value, namely 7, which seemed to give results reasonably close to those obtained with cross-validation on some partial small-scale experiment using the training corpus only.

5.1. Finding threshold

We needed to find an appropriate threshold which would be optimal in some sense for one of the TREC 2002 evaluation measures - either T11SU or T11F. The approach undertaken was to approximate positive and negative datapoints distributions by two - "positive" and "negative" - Gaussians appropriately and then use a Monte-Carlo method to synthesize a new set of data with the same proportion of positive and negative examples as in the training corpus. Further, a threshold "optimal" in terms of the needed evaluation measure from the generated data was induced (Fig. 5.1). The Gaussians means $m_\pm$ and standard deviations $\sigma_\pm$ were estimated from appropriate sets of positive and negative γ_i's.

The pseudocode is given below:

```
Input: $\gamma_i, i = 1,..,M$
    Distances of training documents from the separating hyperplane.
Output: threshold $b$
$b$ = function Two_Gaussian_find_threshold($\{\gamma_i\}$);
    $N_+ = |\{\gamma_i : \gamma_i > 0\}|$;    $N_- = |\{\gamma_i : \gamma_i < 0\}|$;
    $(m_+, \sigma_+)$ = mean_and_std_dev($\{\gamma_i : \gamma_i > 0\}$);
    $(m_-, \sigma_-)$ = mean_and_std_dev($\{\gamma_i : \gamma_i < 0\}$);
    $\Gamma_+$ = sample_normal($m_+, \sigma_+$, 5000 $\frac{N_+}{N_-}$ points);
    $\Gamma_-$ = sample_normal($m_-, \sigma_-$, 5000 $(1 - \frac{N_+}{N_-})$ points);
        $b = \arg\max_{\gamma_i^+ \in \Gamma_+} T11SU(\gamma_i^+, \Gamma_+, \Gamma_-)$;
or $T11F$
end
```

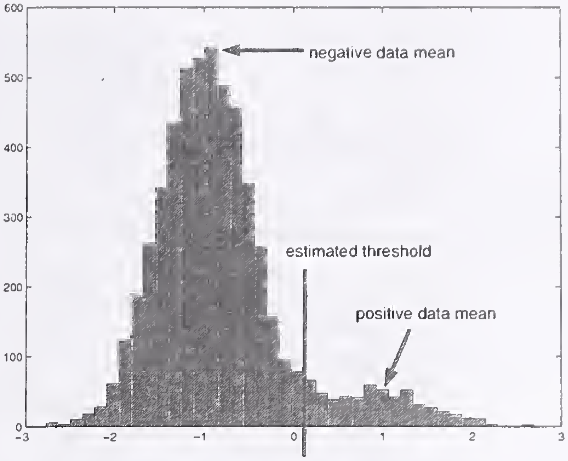


Figure 5.1. An illustration of the method that was chosen to find a threshold value.

Here function $T11SU(\gamma_i^+, \Gamma_+, \Gamma_-)$ computes the T11U score for the threshold γ_i^+ given predictions for the "relevant" Γ_+ and "irrelevant" Γ_- sets of documents.

5.2. TREC results

Results for the TREC batch and routing tasks are shown in Table 3.

The comparative analysis reveals that the method performs relatively well w.r.t. other submissions on the first 50 (assessor) topics whilst on the intersection topics its performance is rather uneven.

Table 4 displays the number of topics on which the method exhibited the best, the 2nd best, etc. result compared to the other 15 participants. Intersection topics turned out to be harder for the Two_Gaussian than for other methods.

6. Why are intersection topics so hard?

Intersection topics turned out to be extremely difficult, both in the adaptive and in the batch settings. In order to gain some insight on the reasons for this, some additional experiments were run after the TREC conference. For six of the fifty intersection topics, representative of different category sizes, the *intersecting* Reuters categories from which they were obtained were identified. These were the following:

run ID	Averaged over topics	T11SU	T11F	AvgP
KerMITT11bf2/rr2	Assessor topics	0.505	0.495	0.427
	Intersection topics	0.245	0.101	0.061
KerMITT11bf2 std. SVM thresh. selection	Assessor topics	0.362	0.121	-
	Intersection topics	0.330	0.010	-

Table 3. Results for the batch and routing tasks. The bottom part contains the results achieved when the "standard" SVM threshold selection is adopted instead of the proposed one.

	best	2nd best	3rd best	2nd worst	worst
# Assessor topics	16	10	8	1	0
# Intersection topics	11	4	9	11	9

Table 4. Relative performance for assessor and for intersection topics.

Query	Categories	
R151	C31	GSCI
R153	C11	M143
R156	M14	E513

Query	Categories	
R157	C331	GPOL
R199	C31	E13
R200	C41	GOBIT

SVMs were trained independently for each of the intersecting Reuters categories, using the threshold selection mechanism described in Section 5, and tested using the TREC split. We computed the performance of the "intersection classifier" obtained by combining the classifiers for the intersecting categories using a logical AND. Results are presented in Table 5. In all cases, the performance of the "intersection classifier" is largely inferior to what one would expect from the performance on the intersecting Reuters categories. This shows that the hindsight provided by the query composition does not help in designing a better classifier for the query, even though the classifiers for each components have relatively good performance. This suggest that the documents in the intersection are atypical in at least one of the intersecting categories.

In order to verify this, we investigated the distribution of the output of the SVM classifers, $f(x) = \sum \alpha_i y_i K(x_i, x)$. By analogy with the large margin argument, we call this the "margin" of an example. In Figure 6.1, we display, for each category, the distribution of the margins of the documents that are 1/ in the intersection (dashed) and 2/ in this category, but not in the intersection (solid). For a perfect classifiers, all margins should be on the right-hand side of the threshold (dotted)—margins on the left-hand side indicate misclassified examples. Figure 6.1 shows that in most cases, the distribution of margins for intersection documents is shifted towards the left, indicating that the intersection documents tend to be misclassified, or in other words, that these documents are either "atypical" for the category, or, at least, harder to learn for the SVM categoriser. A more speculative conjecture would be that intersections contain mostly annotation errors. For query R151, for example, 22 relevant documents represent only 1% of category GSCI, and about 0.06% of C31.

7. Conclusions

The algorithms developed in the context of the KerMIT project seem to perform relatively well on the TREC filtering benchmark. Nevertheless, at least a couple of points remain to be clarified. The first is the very uneven performance on the assessor topics and on the intersection topics. Despite falling short of providing an exhaustive explanation, our experiments show at least that documents relevant to intersection topics tend to be peripheral within the intersecting categories. The second point is the lack of improvement in performance when more complex kernels, such as polynomial kernels of degree higher than one or radial basis function kernels, are used. This is somewhat in contradiction with previous findings in document categorisation (Joachims 1997), which indicated that such kernels do indeed perform better than the basic inner product.

8. Acknowledgments

The authors would like to thank Eric Gaussier, Jaz Kandola, Craig Saunders, and Yoram Singer for the many useful suggestions. The work described in this paper has been supported by the European Commission through the IST Programme under Contract IST-2000-25431 (KerMIT).

References

Cesa-Bianchi, N., A. Conconi, and C. Gentile (2002). A second-order perceptron algorithm. In *Proceedings of the 5th Annual Conference on*

R151	C31		GSCI		C31 AND GSCI	
relevant	20015	16186	1265	923	0	22
irrelevant	11958	674982	225	720728	0	723119
T11F	0.6099		0.7763		0	
R153	C11		M143		C11 AND M143	
	positive	negative	positive	negative	positive	negative
relevant	9265	12292	18254	1376	0	37
irrelevant	8407	693177	2621	700890	33	723071
T11F	0.5022		0.8850		0	
R156	E513		M14		E513 AND M14	
relevant	1503	565	70382	5750	7	65
irrelevant	351	720722	4973	642036	7	723062
T11F	0.7924		0.9321		0.2734	
R157	C331		GPOL		C331 AND GPOL	
relevant	482	596	40856	9726	3	34
irrelevant	143	721920	20437	652122	10	723094
T11F	0.6736		0.6907		0.1685	
R199	C31		E13		C31 AND E13	
relevant	20015	16186	4680	922	35	80
irrelevant	11958	674982	1489	716050	185	722841
T11F	0.6099		0.7728		0.1759	
R200	C41		GOBIT		C41 AND GOBIT	
relevant	8308	1770	87	684	7	79
irrelevant	2559	710504	24	722346	19	723036
T11F	0.7758		0.3580		0.1842	

Table 5. Confusion matrices and T11F scores for six pairs of intersecting classifiers and their intersections.

Computational Learning Theory, LNAI 2375, pp. 121–137. Springer.

Cesa-Bianchi, N., A. Conconi, and C. Gentile (2003). Margin-based algorithms for information filtering. In *Advances in Neural Information Processing Systems 15*. MIT Press.

Cristianini, N., J. Shawe-Taylor, and H. Lodhi (2002). Latent semantic kernels. *Journal of Intelligent Information System 18*(2/3), 127–152.

Hoerl, A. and R. Kennard (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics 12*, 55–67.

Joachims, T. (1997). Text categorization with support vector machines: Learning with many relevant features. Technical report, Universitaet Dortmund.

Joachims, T. (2002). SVM^{light} - Support Vector Machine. http://svmlight.joachims.org.

Lewis, D. D. (2001). Applying support vector machines to the trec-2001 batch filtering and routing tasks. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pp. 286–292.

Li, Y., H. Zaragoza, R. Herbrich, J. Shawe-Taylor, and J. Kandola (2002). The perceptron algorithm with uneven margins. In *Proceedings of ICML 2002*, pp. 379–386.

Vapnik, V. (1998). *Statistical Learning Theory.* Chichester, UK: Wiley.

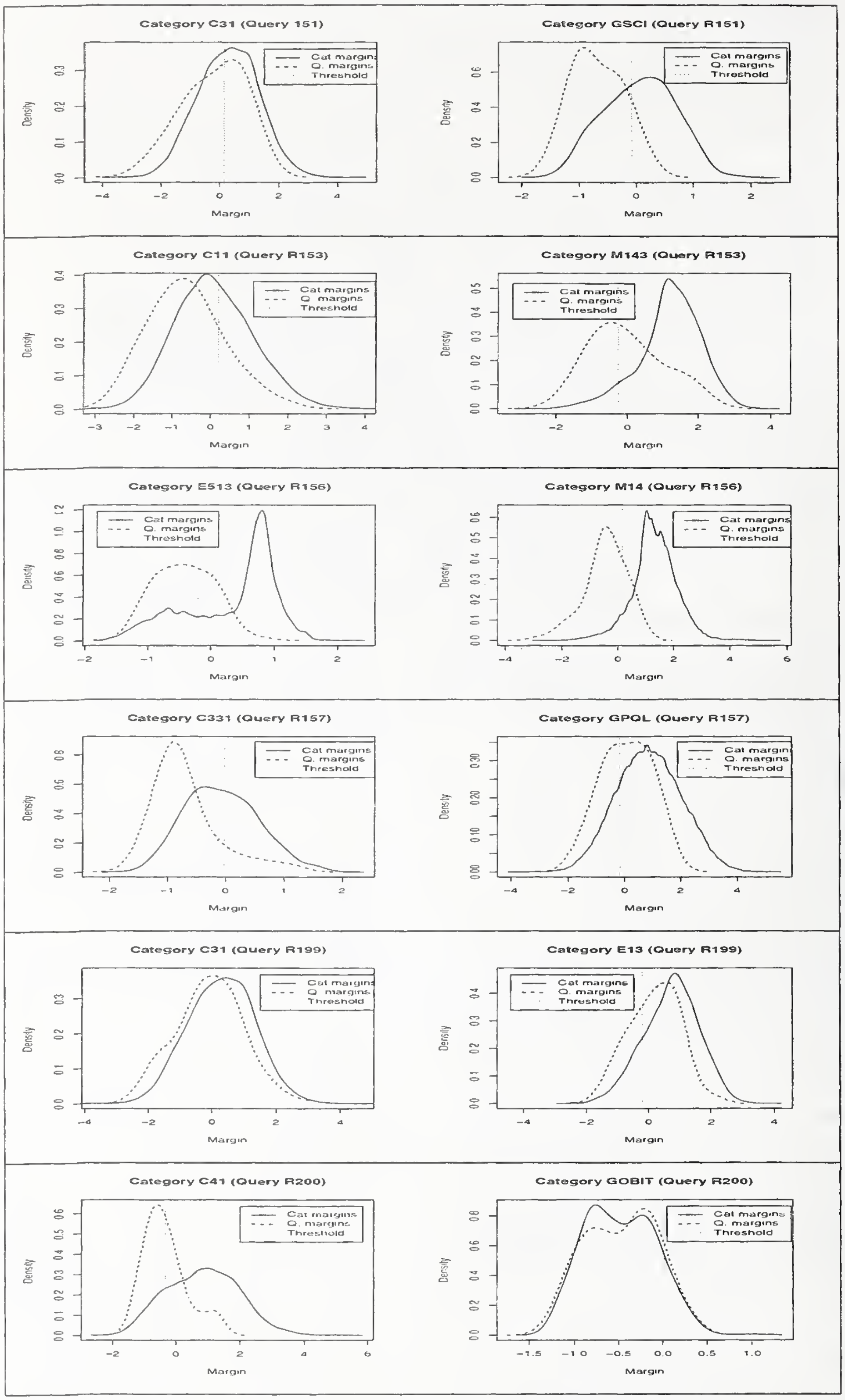


Figure 6.1. Distributions of the margins of the documents within the intersections (dashed lines) and within the category but out of the intersection (solid lines) with respect to the hyperplanes trained independently for the intersecting categories (dotted line).

LIT at TREC-2002: Web Track

Nie Yu, Ji Donghong, Yang Lingpeng
Laboratories for Information Technology, Singapore
{ynie, dhji, lpyang}@lit.a-star.edu.sg
http://www.lit.org.sg

Abstract

In Trec-2002, we participated in the Web Trec (named page finding task). There are two kinds of information that can be used while finding the expected page, content information and link information. We exploited both of them. That is to say, our system is content-based and link-based. As to link information, we only used anchor text and connections, and topology between pages is ignored. We submitted two runs. One is based on traditional contented-based retrieval, the other try to combine content-based retrieval and link-based retrieval to get better result.

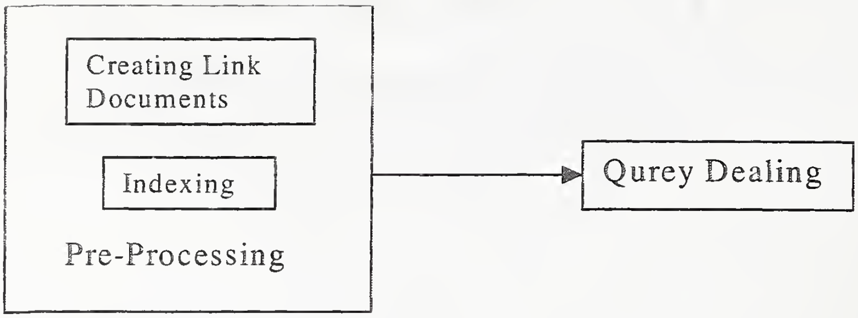
1 Introduction

This is the first time we participate in Web Trec. We focused our work on named page finding task. To exploit link information more explicitly, we extracted link information of each page that is used to create the link document (Craswell, et al., 2001; Dumais, 2001; Savoy, 2001). So each page has its respective link document.

For each query, we extracts source terms from the query, creates vectors for each web page and its link document according to source terms, and then calculates evaluating value of each web page (Robertson et al., 2001). Finally we get the ranked page list via sorting evaluating values of web pages.

2 System Description

We did the pre-processing work before dealing with queries. The process is as the diagram below.



We did the pre-processing work with three desktop computers (750MHz and 128M memory, Windows 98). Queries dealing runs on one desktop computer (750MHz, 512M memory, Windows 98).

3 Pre-processing

Before dealing with queries, we created link document for each web page, and indexed all web pages. The glossary and stop word list are from Wordnet.

After pre-processing, we got two tables and indexed result. The title&anchor table contains titles and anchor descriptions of web pages. The link table denotes which pages are pointing to a given page.

3.1 Creating Link Documents

For each web page d, a link document ld is created. The link document ld contains all anchor text of hyperlinks which pointing to web page d in other pages. The sum of those hyperlinks is recorded as sum_d. We defined the title of a web page as the anchor text of one additional hyperlink, because we thought title should be similarly accurate with pointing-in hyperlinks on indicating a page's meaning.

3.2 Indexing

For each web page, all words were stemmed and stop words were removed before indexing.

4 Named Page Finding

First, we extract source terms from the query, and then we built context vectors and link document vectors for pages based on source terms. By comparing the evaluating value of all pages, we get the final result sequence.

4.1 Extracting Source Terms

All stop list words are ignored before source term extracting. For one query q such as w_1 w_2 $w_3 \ldots$ w_n, w_i is a single word, $1 \leq i \leq n$, there are n(n-1)/2 possible source term t.

$$t \varepsilon T$$

$$T = \{ w_j \, w_{j+1} \ldots w_{j+l} \,|\, 1 \leq j \leq n, 1 \leq l \leq n-j\} \cup \{ w_j \,|\, 1 \leq j \leq n\}$$

Each source term t is given a weight value wt_t:

$$wt_t = F(l_t, l_q)$$

where l_t is the length of source term t, l_q is the length of query q. So for source terms T, a weight vector wt is created.

$$wt = (wt_{t1}, wt_{t2}, \ldots, wt_{tm}), m=n(n-1)/2$$

4.2 Creating Context Vector and Link Document Vector

We create a context vector v and a link document vector vl for each web page d regarding source terms T.

$$v = (f_1, f_2, \ldots, f_{n(n-1)/2})$$

$$vl = (fl_1, fl_2, \ldots, fl_{n(n-1)/2})$$

where f_i ($1 \leq i \leq n(n-1)/2$) is the frequency of t_i ($1 \leq i \leq n(n-1)/2$) in web page d, fl_i ($1 \leq i \leq n(n-1)/2$) is the frequency of t_i ($1 \leq i \leq n(n-1)/2$) in link document ld. Frequencies will not be repeatedly counted for different terms. Longer terms have priority over shorter terms.

4.3 Evaluating Value

For each web page d, an evaluating value e is calculated following the equation below:

$$e = f(v/f1(l_d), wt) + \mu f(vl/f2(sum_d), wt)$$

where f,f1,f2 are functions, μ is a parameter to weight effect of link information, which is valued after a lot of tests.

In fact, we delivered two runs, one is content-based, and the other is content & link-based. In the content-based run, we used equation below:

$$e = f(v/f1(l_d), wt)$$

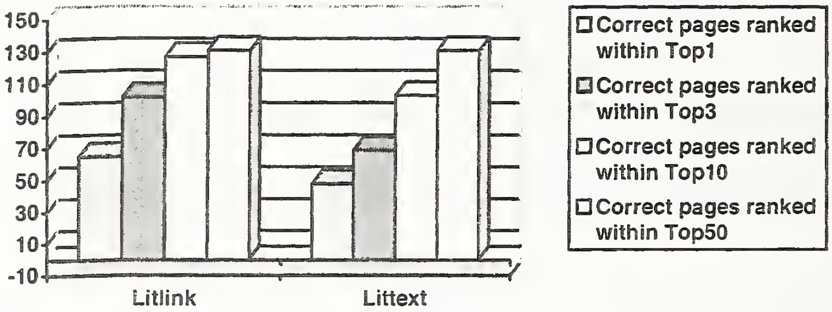
4.4 Get Ranked Page List

After comparing and sorting evaluating value e of each web page, we can get the ranked page list.

5 Result

We submitted 2 runs for Web Trec task. The results are as Chart 1 below:

Chart 1



In the chart, the value of "top n" column means how many named pages were found in the top n ranked results within 150 given queries. The run 'litlink ' used both content information and link information, run 'littext' only used content information. The accuracy of two runs is compared in the table below. We can easily find out that with link information, much better results are obtained.

Run Id	**Top 1**	**Top 3**	**Top 10**	**Top 50**
Litlink	43.3%	68%	84.7%	87.3%
Littext	32%	46%	68.7%	87.3%

6 Conclusion and Future Work

Unlike the set of plain text documents, web pages are document set with topology structure and relationships. The titles and anchor descriptions actually provide the summary indicating kernel content of pages. They can remarkably improve the efficiency and performance of retrieval, should and must be considered in retrieval process.

We will continue focusing on retrieval based on link information. Different process should be taken to consider different kind of link information e.g., title, anchor.

In addition, we will try to combine content-based retrieval and link-based retrieval better to achieve higher performance.

References

Craswell, N., Hawking, D., and Robertson, S.E.(2001). Effective site finding using link anchor information. In SIGIR-01.

Dumais, S., and Jin, R.(2001). Probalistic combination of content and links. In SIGIR-01.

Savoy, J., and Rasolofo, Y. (2000). Report on the TREC-9 Experiment: Link-Based Retrieval and Distributed Collections. In TREC-9.

Robertson, S.E, Spark-Jones K., Relevance weighting of search terms. In Journal of ASIS, 27, pp.129-146, 1976

LCC Tools for Question Answering

Dan Moldovan, Sanda Harabagiu, Roxana Girju, Paul Morarescu, Finley Lacatusu,
Adrian Novischi, Adriana Badulescu and Orest Bolohan

Language Computer Corporation
Richardson, TX 75080
email: moldovan@languagecomputer.com

Abstract

The increased complexity of the TREC QA questions requires advanced text processing tools that rely on natural language processing and knowledge reasoning. This paper presents the suite of tools that account for the performance of the PowerAnswer question answering system. It is shown how questions, answers and world knowledge are transformed first in logic representation, followed by a systematic and rigorous logic proof that validly answers questions posed to the QA system. At TREC QA 2002, PowerAnswer obtained a confidence-weighted score of 0.856, answering correctly 415 out of 500 questions.

1 Introduction

To cope with the continuously increasing difficulty of the TREC QA task, LCC has enhanced the processing capability of the QA system by improving some modules and strengthening the set of tools involved in answer extraction. This paper presents a general overview of the main tools and focuses on some of them.

PowerAnswer, the QA system developed at LCC, searches for answers from large collections of texts by combining syntactic, semantic, lexical and world knowledge information sources. PowerAnswer consists of three main parts: question processing, document retrieval and answer extraction. In turn, each part consists of smaller modules that work collectively to produce question answers. The relative performance of these modules was described in [Moldovan et al 2002].

Questions and relevant document paragraphs are parsed and transformed into logic forms that together with world knowledge axioms extracted from the WordNet glosses are fed to a logic prover. For each question, the result is a set of ranked answers accompanied by their measures of beliefs.

Advanced QA requires sophisticated text processing tools based on the state-of-the-art NLP and reasoning methods. The LCC tool set includes: Name Entity Recognizer, Syntactic Parser, Logic Form Transformer, Word Sense Desambiguator, Lexical Chainer, Logic Prover and others. In addition, since PowerAnswer operates in comercial environments, it has a set of support tools such as System Manager, PowerAnalytics, PowerIndex, Power Ontology, and Document Manager that enhance its functionality. The main focus of this paper is on Lexical Chainer and Logic Prover.

2 Name Entity Recognizer

Name Entity Recognition systems identify named-entities such as people, organizations, dates, places, quantities and others in text documents. LiteNE, the LCC Name Entity Recognizer is implemented as a cascade of finite-state automata interleaved with other preprocessing and coreference resolution.

The *tokenizer* is the first module of the LiteNE system. The task of the tokenizer is to break the document into lexical entities (tokens). Generally, a token is a word or a punctuation sign, but in some cases it may be a word fragment. For example, the word "U.S.A." is broken into six tokens: "U", ".", "S", ".", "A", and ".". It is the task of the next module to identify the above sequence as a complex word representing an abbreviation of "United States of America".

The *Lexicon* module identifies words and complex words that have lexical and semantic meaning. This is done by inspecting both dictionaries and gazetteers. Dictionaries contain open-domain lexico-semantic information, e.g. "house" is an artifact, or to a lesser extent domain-specific information, e.g. "mail bomb" is a kind of bomb. Gazetteers typically store well-known entity names, such as locations, e.g. "Dallas" is a city in the state of "Texas" part of the country "United States of America".

The *Preprocessor* identifies simple lexical entries that are not stored in lexicon or gazetteers. Some of the items identified by the preprocessor are: phone numbers "1-(800) 888-7777", money "$4.75", dates "December 25", times "8:30am", measures such as "10kg", "one hundred degrees", and others.

The core *Name Entity Recognizer* assigns lexical features to words or groups of words such as locations, organizations, persons, addresses, and others. Proper names are particularly useful for extraction systems since they point to objects about which we need to identify properties, relations, events. The technique is to use capitalization if available. Some of the most frequently used methods are Hidden Markov Models and finite state automata patterns. With the help of dictionaries these techniques are able to recognize that "John Smith" is a proper name, and "John Hopkins" is a University; or that "Austin Ventures" is a company, "Austin, Texas" is a city and "Austin Thomas" is a name. Machine learning methods were used to train the LiteNE system. NE-recognition benefits by morphological analysis by looking up in a dictionary for all morphological variations of words.

Part of Speech Tagging is useful for subsequent text analysis stages. This involves specifying the part of speech of each word. POS tagger combines rule-based and statistical methods and achieves an accuracy around 96%.

3 Syntactic Parser

In advanced QA, like in many other NLP applications, syntactic parsing plays a major role in the overall system accuracy, especially since subsequent steps rely on it. In the last few years, LCC developed its own parser, and a version was trained for QA, meaning it has the capability of parsing questions as well as free text. It is a probabilistic parser which has been improved over the years.

Parser identifies simple noun phrases ("the fast red car"), verb phrases ("is being obverved daily"), and also particles that may be significant in subsequent text analysis. It recognizes phrases and solves the attachment of prepositional phrases and close subordination. Full syntactic parsing takes long time to process even a small number of text documents. Since we reduce documents to relevant passages we can afford to fully parse these passages. The quality of parser affects the accuracy of subsequent steps.

Coreference Resolution is the task of determining that a noun phrase refers to the same entity as another noun phrase. This involves equating various forms of personal proper names, for example "President Bush", "George Bush", "the 43rd President of US", etc. There are other more complex forms of coreference such as definite or indefinite noun phrase and pronoun coreference that have been implemented. Also, a light form of temporal coreference resolution has been implemented.

4 Logic Form Representation

The logic form (LF) is an intermediary step between syntactic parse and the deep semantic form. The LF codification acknowledges syntax-based relationships such as: (1) syntactic subjects, (2) syntactic objects, (3) prepositional attachments, (4) complex nominals, and (5) adjectival/adverbial adjuncts.

There are two criteria that guide our approach: (1) the notation be as close as possible to English, and (2) the notation be syntactically simple. Our approach is to derive the *LF directly from the output of the syntactic parser.* The parser resolves the structural and syntactic ambiguities.

The basis of integrating a Logic Form representation into the PowerAnswer system is that all questions and relevant paragraphs are transformed into an unambiguous logic representation. The term Answer Logic Form (ALF) refers to the candidate answers in logic form. Candidate answers returned by the Answer Extraction module are classified as free text due to the unpredictable nature of their grammatical structure. The term Question Logic Form (QLF) refers to the questions posed the Question Answering system in logic form.

Essentially there is a one to one mapping of the words of the text into the predicates in the logic form. The predicate names consist of the base form of the word concatenated with the part of speech of the word. Each noun has an argument that is used to represent it in other predicates. One of the most important features of the Logic Form representation is the fixed-slot allocation mechanism of the verb predicates. This allows for the Logic Prover to see the difference between the role of the subjects and objects in a sentence that is not answerable in a keyword based situation.

Logic Forms are derived from the grammar rules found in the parse tree of a sentence. There are far too many grammar rules in the English language to efficiently and realistically implement them all. We have observed that the top ten most frequently used grammar rules cover 90% of the cases for WordNet glosses. This is referred to as the 10-90 rule. Below we provide a sample sentence and its corresponding LF representation. More details regarding the transformation of text into logic forms are presented in [Moldovan and Rus 2001].

Example:
Heavy selling of Standard & Poor's 500-stock index futures in Chicago relentlessly beat stocks downward.

LF:
heavy_JJ(x1) & selling_NN(x1) & of_IN(x1,x6) & Standard_NN(x2) & &_CC(x13,x2,x3) & Poor_NN(x3) &'s_POS(x6,x13) & 500-stock_JJ(x6) & index_NN(x4) & future_NN(x5) & nn_NNC(x6,x4,x5) & in_IN(x1,x8) & Chicago_NN(x8) & relentlessly_RB(e12) & beat_VB(e12,x1,x9) & stocks_NN(x9) & downward_RB(e12)

5 Lexical Chains

A major problem in QA is that often an answer is expressed with words different from the question keywords. In such cases it is useful to find topically related words to the question keywords. By exploiting the information in the WordNet glosses, the connectivity between the synsets is dramatically increased. When a word in a gloss is semantically disambiguated, it points to the synset it belongs to. We call this extended WordNet (XWN) [Harabagiu, Miller and Moldovan 1999]. In the context of XWN, or any other lexical database, topical relations can be expressed as *lexical chains.* These are sequences of semantically related words that link two concepts. Lexical chains have been used in computational linguistics to study: discourse, coherence, inference, implicatures malapropisms and others [Morris and Hirst 1991], [Hirst and St-Onge 1998], [Harabagiu and Moldovan 1998b]. Lexical chains improve the performance of question answering systems in two ways: (1) increase the document retrieval recall and (2) improve the answer extraction by providing the much needed world knowledge axioms that link question keywords with answers concepts.

It is possible to establish some connections between synsets via topical relations. We developed software that automatically provides connecting paths between any two WordNet synsets S_i and S_j up to a certain distance. The meaning of these paths is that the concepts along a path are topically related.

In Table 1 we show a few examples of morphologically related words that appear in synsets and their glosses which are brought to bear by the topical relations.

Synset	**Gloss**	**Morphological relation**
laughter:n#1	(the sound of laughing:v#1)	noun - verb
immediately:r#3	(bearing an immediate:a#2 relation)	adverb - adjective
insure:v#4	(take out insurance:n#3 for)	verb - noun
parental:adj#2	(..characteristic of or befitting a parent:n#1)	adjective - noun

Table 1: Examples of new morphological relations revealed by the topical relations

Examples

Below we provide the most relevant lexical chains that link some selected TREC 2002 questions with their answers.

Q1403: When was the internal combustion engine invented ?
Answer: The first internal - combustion engine was built in 1867
Lexical chains:
(1) invent:v#1 → HYPERNYM → create_by_mental_act:v#1 → HYPERNYM → create:v#1 → HYPONYM → build:v#1

Q1404: How many chromosomes does a human zygote have ?
Answer: 46 chromosomes that lie in the nucleus of every normal human cell
Lexical chains:
(1) zygote:n#1 → HYPERNYM → cell:n#1
(2) zygote:n#1 → HYPERNYM → cell:n#1 →HAS_PART → nucleus:n#1

Q1411: What Spanish explorer discovered the Mississippi River ?
Answer: Spanish explorer Hernando de Soto reached the Mississippi River
Lexical chains:
(1) discover:v#7 → GLOSS → reach:v#1

Q1462: Where is the oldest synagogue in the United States ?
Answer: Newport is marking the 350th anniversary of the founding of Trinity Church , and is also home to the nation 's oldest synagogue
Lexical chains:
(1) United_States:n#1 → HYPERNYM → North_American_country:n#1 → HYPERNYM → country:n#1 → GLOSS → nation:n#1

Q: 1518 What year did Marco Polo travel to Asia ?
Answer: Marco Polo divulged the truth after returning in 1292 from his travels , which included several months on Sumatra.
Lexical chains:
(1) travel_to:v#1 → GLOSS → travel:v#1 → RGLOSS → travel:n#1
(2) travel_to#1 → GLOSS → travel:v#1 → HYPONYM → return:v#1
(3) Sumatra:n#1 → ISPART → Indonesia:n#1 → ISPART → Southeast_Asia:n#1 → ISPART → Asia:n#1

Q: 1540 What is the deepest lake in America ?
Answer: Rangers at Crater Lake National Park in Oregon have closed the hiking trail to the shore of the nation 's deepest lake
Lexical chains:
(1) America:n#1 → HYPERNYM → North_American_country:n#1 → HYPERNYM → country:n#1 → GLOSS → nation:n#1

6 Logic Prover

Usefulness of a Logic Prover in Question Answering

The LCC Logic Prover renders a deep understanding of the relationship between the question text and answer text. The Logic Prover captures the syntax-based relationships such as the syntactic objects, syntactic subjects, prepositional attachments, complex nominals, and adverbial/adjectival adjuncts provided by the LF representation. In addition to the LF representations of questions and candidate answers, the Logic Prover needs world knowledge axioms to link questions to answers. For this, the Logic Prover uses the Lexical Chains to bring to the forefront the most important logic axions needed in a proof. In XWN, an axiom is the LF expression of a synset and its gloss. With this deep and intelligent representation, the Logic Prover effectively and efficiently re-ranks candidate answers by their correctness and ultimately eliminates incorrect answers. In this way, the Logic Prover is a powerful tool in boosting the accuracy of the PowerAnswer system. Moreover, the trace of a proof constitutes a justification for that answer.

LCC's Logic Prover

The base of LCC's Logic Prover is Otter, an automated reasoning system developed at Argonne Labs. Extensions were made to customize Otter to the Question Answering task. The inference rule sets are based on hyperresolution and paramodulation. Hyperresolution is an inference rule that does multiple binary resolution steps in one, where binary resolution is an inference mechanism that looks for a positive literal in one clause and negative form of that same literal in another clause such that the two literals can be canceled, resulting in a newly inferred clause. Paramodulation introduces the notion of equality substitution so that axioms representing equality in the proof do not need to be explicitly included in the axiom lists. Additionally, similar to hyperresolution, paramodulation combines multiple substitution steps into one.

The search strategy used is the Set of Support Strategy, which partitions the axioms used during the course of a proof into those that have support and those that are considered auxiliary. The axioms with support are placed in the Set of Support (SOS) list and are intended to guide the proof. The auxiliary axioms are placed in the Usable list and are used to help the SOS infer new clauses. This strategy restricts the search such that a new clause is inferred if and only if one of its parent clauses come from the Set of Support. The axioms that are placed in the SOS are the candidate answers, the question negated (to invoke the proof by contradiction), axioms related to linking named entities to answer types, and axioms related to decomposing conjunctions, possessives, and complex nominals. Axioms placed in the Usable list are the WordNet axioms and other axiom based outside world knowledge.

The Logic Prover will continue trying to find a proof until one of two conditions is met; either the Set of Support becomes empty or a refutation is found.

Examples of answer justification in action:

Example 1.

Question 1797: How did Adolf Hitler die ?
QLF: manner_AT(e1) & adolf_nn(x2) & hitler_nn(x3) & nn_nnc(x4,x2,x3) & die_vb(e1,x4,x1)

Question Axiom:
-(exists e1 x1 x2 x3 x4 (adolf_nn(x2) & hitler_nn(x3) & nn0_nnc(x4,x2,x3) & die_vb(e1,x4,x1))).

Answer:
It was Zhukov 's soldiers who planted a Soviet flag atop the Reichstag on May 1 , 1945 , a day after Adolf Hitler committed suicide.

We introduce a psuedo-verb for suicide since the WordNet gloss for the noun suicide lets us infer that suicide is an act and therefore can be treated as a verb.

ALF:
It_PRP(x14) & be_VB(e1,x14,x2) & Zhukov_NN(x1) & 's_POS(x2,x1) & soldier_NN(x2) & plant_VB(e2,x2,x3) & Soviet_JJ(x3) & flag_NN(x3) & atop_IN(e2,x4) & Reichstag_NN(x4) & on_IN(e2,x8) & May_NN(x5) & 1_NN(x6) & 1945_NN(x7) & nn_NNC(x8,x5,x6,x7) & day_NN(x9) & Adolf_NN(x10) & Hitler_NN(x11) & nn_NNC(x12,x10,x11) & commit_VB(e3,x12,x13) & suicide_NN(x13) & suicide_VB(x13,x19,x12)

Answer Axiom:
exists e1 e2 e3 e4 x1 x10 x11 x12 x13 x14 x17 x18 x19 x2 x3 x4 x5 x6 x7 x8 x9 (it_prp(x14) & be_vb(e1,x14,x2) & zhukov_nn(x1) & _s_pos(x2,x1) & soldiers_nn(x2) & planted_vb(e2,x2,x3) & soviet_jj(x3) & flag_nn(x3) & atop_in(e2,x4) & reichstag_nn(x4) & on_in(e2,x8) & may_nn(x5) & 1_nn(x6) & 1945_nn(x7) & nn_nnc(x8,x5,x6,x7) & day_nn(x9) & adolf_nn(x10) & hitler_nn(x11) & nn_nnc(x12,x10,x11) & commit_vb(e3,x12,x13) & suicide_nn(x13) & suicide_vb(x13,x19,x12)).

Wordnet Relations:
Suicide is a manner of killing.
Suicide_NN(e1) → kill_NN(e1) & manner_AT(e1)
Axiom:
all e1 (suicide_nn(e1) → kill_nn(e1) & manner_at(e1)).

Pseudo-Verb Wordnet Gloss:
Suicide is the act of killing yourself
Gloss Logic Form:
suicide_VB(e1,x1,x2) ↔ kill_VB(e2,x1,x2) & yourself_PRP(x2) *Axiom:* all e1 x1 x2 (suicide_vb(e1,x1,x2) → kill_vb(e1,x1,x2) & yourself_nn(x2)).

Wordnet Gloss:
To kill is to cause to die
Gloss Logic Form:
kill_VB(e1,x1,x2) ↔ cause_VB(e2,x1,e3) & die_VB(e3,x2,x4)
Axiom:
all e1 e2 e3 x1 x2 x4 (kill_vb(e1,x1,x2) ↔ cause_vb(e2,x1,e3) & die_vb(e3,x2,x4)).

Linguistic Axioms:
Link the noun kill to the verb kill
all e1 e2 x1 x2 (kill_nn(e1) & kill_vb(e2,x1,x2) → kill_vb(e1,x1,x2)).

Make the yourself predicate relfexively used in the verb kill
all e1 x1 x2 (kill_vb(e1,x1,x2) & yourself_nn(x2) → yourself_nn(x1)).

The relevent steps in the proof:
1 [] -manner_at(x15) | -adolf_nn(x2) | -hitler_nn(x3) | -nn_nnc(x4,x2,x3) | -die_vb(x15,x4,x1).
(The question negated to invoke a proof by contradiction)
18 [] adolf_nn($c16).
19 [] hitler_nn($c15).
20 [] nn_nnc($c14,$c16,$c15).
22 [] suicide_nn($c13).
23 [] suicide_vb($c13,$c9,$c14).
(The Logic Prover selects the above clauses from the answer)
25 [] -suicide_nn(x16) | manner_at(x16).
(The Logic Prover selects the axiomatic knowledge extracted from Wordnet that suicide is a manner of killing)
29 [] -kill_vb(x23,x1,x2) | die_vb(x23,x2,$c23).
(The Logic Prover selects the WordNet gloss for kill implies die)
30 [] -suicide_vb(x24,x1,x3) | kill_vb(x24,x1,x3).
(The Logic Prover selects the WordNet gloss for suicide implies kill) 32 [hyper,22,25] manner_at($c13).
35 [hyper,23,30] kill_vb($c13,$c9,$c14).
36 [hyper,35,29] die_vb($c13,$c14,$c23).
39 [hyper,1,32,18,19,20,36] $F.

In the final step the terms for adolf_nn($c16), hitler_nn($c15), nn_nnc($c14,$c16,$c15), manner_at($c13), and die_vb($c13,$c14,$c23) are hyperresolved with the negated question to yield a full proof by contradiction.

Example 2.

Question 1512: What is the age of our solar system ?

QLF:
_quantity_AT(x2) & age_NN(x2) & of_IN(x2,x3) & solar_JJ(x3) & system_NN(x3)

Question Axiom:
-(exists x1 x2 x3 (_quantity_at(x2) & age_nn(x2) & of_in(x2,x3) & solar_jj(x3) & system_nn(x3))).

Answer:
The solar system is 4.6 billion years old

ALF:
solar_JJ(x5) & system_NN(x5) & 4.6_NN(x2) & billion_NN(x3) & year_NN(x4) & nn_NNC(x5,x2,x3,x4) & old_JJ(x5)

Answer Axiom:
exists e1 x1 x2 x3 x4 x5 x7 (solar_jj(x5) & system_nn(x5) & 4_6_nn(x2) & billion_nn(x3) & years_nn(x4) & nn_nnc(x5,x2,x3,x4) & old_jj(x5)).

Wordnet Gloss:
Old is having lived for a relatively long time or attained a specific age.

Gloss Logic Form:
old_JJ(x6) ↔ live_VB(e2,x6,x2) & for_IN(e2,x1) & relatively_JJ(x1) & long_JJ(x1) & time_NN(x1) & or_CC(e5,e2,e3) & attain_VB(e3,x6,x2) & specific_JJ(x2) & age_NN(x2)

Axiom:
all e2 e3 e5 x1 x2 x6 (old_jj(x6) ↔ live_vb(e2,x6,x2) &for_in(e2,x1) & relatively_jj(x1) & long_jj(x1) & time_nn(x1) & or_cc(e5,e2,e3) & attain_vb(e3,x6,x2) & specific_jj(x2) & age_nn(x2)).

Named entity axioms:
all x2 x3 x4 x5 (4_6_nn(x2) & billion_nn(x3) & years_nn(x4) & nn_nnc(x5,x2,x3,x4) → _quantity_at(x5)).

Linguistic axioms:
all x1 (_quantity_at(x1) & solar_jj(x1) & system_nn(x1) → of_in(x1,x1)).

The relevent steps in the proof:
1 [] -_quantity_at(x2) | -age_nn(x2) | -of_in(x2,x3) | -solar_jj(x3) | -system_nn(x3).
(The question negated to invoke a proof by contradiction)
2 [] solar_jj($c2).
3 [] system_nn($c2).
4 [] 4_6_nn($c5).
5 [] billion_nn($c4).
6 [] years_nn($c3).
7 [] nn_nnc($c2,$c5,$c4,$c3).
8 [] old_jj($c2).
(The Logic Prover selects the above clauses from the answer)
17 [] -old_jj(x6) | age_nn(x2).
(The Logic Prover selects the WordNet gloss for old implies age)
19 [] -4_6_nn(x2) | -billion_nn(x3) | -years_nn(x4) | -nn_nnc(x5,x2,x3,x4) | _quantity_at(x5).
(The Logic Prover selects the named entity axiom linking 4.6 billion years to a quantity)
20 [] -_quantity_at(x1) | -solar_jj(x1) | -system_nn(x1) | of_in(x1,x1).
(The Logic Prover selects the Linguistic axiom implying that if a noun or noun phrase is a quantity then the quantity is an attribute of that noun or noun phrase)
21 [hyper,8,17] age_nn(x).
30 [hyper,7,19,4,5,6] _quantity_at($c2).
31 [hyper,30,20,2,3] of_in($c2,$c2).
32 [hyper,1,30,21,31,2,3] $F.

In the final step the terms for _quantity_at($c2), of_in($c2,$c2), age_nn(x), solar_jj($c2), and system_nn($c2) are hyperresolved with the negated question to yield a full proof by contradiction.

When the proof fails, we devised a way to incrementally relax some of the conditions that hinder the completion of the proof. This relaxation process puts weights on the proof such that proofs weaker than a predefined threshold are not accepted.

7 Other Support Tools

System Manager provides immediate access to all key parameters and indicators for managing and administering the Question Answering system, e.g. answer rate, number of simultaneous questions, static or dynamic content collection, number of documents retrieved, memory size for Natural Language parsing, and many others. With its web-based, easy to use interface, the System Manager allows for all these parameters to be adjusted in real time. The result is an Answer service that follows the needs of the Customer Service department, scaling up and down for the optimal performance and resource allocation.

PowerAnalytics offers continuous customer feedback with up-to-date insight on customer needs, communication patterns and expectations. The advanced management module with friendly and easy-to-use web interfaces makes real-time service monitoring and content adjustment fast and powerful. Companies benefit from low maintenance costs and deep knowledge of business and customer service metrics in both standard and highly customized analytic reports.

PowerIndex makes content integration fast and easy. The system deploys advanced harvesting tools that access, collect and index large amounts of structured or unstructured information automatically, or on request. New and old knowledge is merged seamlessly and re-indexed most efficiently, so that fresh information is disseminated as soon as it is created.

PowerOntology creates deep concept ontologies starting from only a small number of concept seeds. For example, the WordNet ontology is enhanced with domain specific concepts through syntactic transformations, such as extending the seed concept with modified nouns, and through semantic chains, such as ISA-relations.

Document Manager allows for clear management of all structured and unstructured data. It keeps track of multiple document revisions, add-ons or deletions, data changes and document locations. Document Manager also integrates seamlessly with PowerIndex and offers a clear overview on the content resources and their immediate availability for fresh knowledge dissemination through our Question Answering and Information Extraction products.

8 Results at TREC 2002

The performance obtained by PowerAnswer at TREC QA 2002 in the main task is summarized in Table 2.

9 Conclusions

This paper introduced some of the tools that support the operation of LCC's QA system. In particular, it was demonstrated how questions, document paragraphs and world knowledge axioms are formally represented and how a logic prover methodically and efficiently generates correct answers.

Number wrong (W)	63
Number unsupported (U)	14
Number inexact (X)	8
Number right (R)	415
Precision of recognized no answer	0.578
Recall of recognizing no answer	0.804
Confidence-weighted score	**0.856**

Table 2: Performance over 500 questions

Essential in this framework is the extended WordNet which supplies the prover with world knowledge axioms. To cope with future questions that entail implicatures and other complex analyses, nonmonotonic reasoning methods need to be incorporated into the logic prover. It becomes more and more clear that QA is intimately linked with natural language processing, text mining, and reasoning on knowledge bases.

Acknowledgement

This work was supported in part by the ARDA AQUAINT program. We wish to thank Christine Clark, Mihai Surdeanu and Marius Pasca from LCC for their contribution to this work.

References

[Fellbaum 1998] Christiane Fellbaum. WordNet - *An Electronic Lexical Database*, MIT Press, Cambridge, MA, 1998.

[Harabagiu and Moldovan 1998b] S.M. Harabagiu and D.I. Moldovan. Knowledge Processing on an Extended WordNet. WordNet-An Electronic Lexical Database. The *MIT Press*, C. Fellbaum editor, pp 379-406, 1998.

[Harabagiu, Miller and Moldovan 1999] S. Harabagiu, G.A. Miller, and D.I. Moldovan. WordNet 2 - A Morphologically and Semantically Enhanced Resource. *Proceedings of ACL-SIGLEX99: Standardizing Lexical Resources*, Maryland, June 1999, pp.1-8.

[Hirst and St-Onge 1998] G. Hirst and D. St-Onge. Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms. WordNet-An Electronic Lexical Database. The *MIT Press*, C. Fellbaum editor, pp 305-332, 1998.

[Moldovan and Rus 2001] D. Moldovan and V. Rus. Logic Form Transformation and its Applicability to Question Answering. In *Proceedings of ACL 2001.*

[Moldovan et al 2002] D. Moldovan, M. Pasca, S. Harabagiu and M. Surdeanu. Performance Issues and error Analysis in an Open-Domain Question Answering System. In *Proceedings of ACL 2002.*

[Moldovan and Novischi 2002] D. Moldovan and A. Novischi. Lexical Chains for Question Answering. In *Proceedings of COLING 2002*, pp.674-680.

[Morris and Hirst 1991] J. Morris and G. Hirst. Lexical Cohesion Computed by Thesaural Relations as an Indicator of the Structure of Text. In *Computational Linguistics*, Vol. 17 , no 1, pp. 21-48, 1991.

Coupling Named Entity Recognition, Vector-Space Model and Knowledge Bases for TREC-11 Question Answering Track

P. Bellot[(1)], E. Crestan[(1,2)], M. El-Bèze[(1)], L. Gillard[(1)], C. de Loupy[(2)]

(1) Laboratoire d'Informatique d'Avignon (LIA)
339 ch. des Meinajaries, BP 1228
F-84911 Avignon Cedex 9 (France)
{ patrice.bellot, eric.crestan, marc.elbeze. laurent.gillard }@lia.univ-avignon.fr

(2) Sinequa S.A.S.
51, rue Ledru Rollin
F-94200 Ivry-sur-Seine (France)
{ loupy, crestan }@sinequa.com

Abstract: In this paper, we present a question-answering system combining Named Entity Recognition, Vector-Space Model and Knowledge Bases to validate answers candidates. Applying this hybrid approach, for our first participation in the TREC Q&A.

Keywords: Question Answering, Named Entity Recognition, Vector-Space Model, Knowledge Bases

1. Introduction

Our approach combines a Named Entity Recognition System developed at Sinequa[1] and an answer retrieval system based on Vector Space model that uses some Knowledge Bases developed at the Laboratoire d'Informatique d'Avignon[2].

First, the Named Entity Recognition system is briefly described, including specific features (section 2). Then, a summarized description of the SIAC (Segmentation et Indexation Automatique de Corpus) information retrieval system is given (section 3). For the purpose of Question Analysis, several Question Taggings have been employed, they are exposed in section 4. The approach using Knowledge Bases is then depicted (section 5), with a summary of its coverage (section 5.3). Section 6 is devoted to the Question Ordering problem. Finally, we present several experiments in the frame of TREC-11 (section 7).

2. Named Entities

Detection of Named Entities (NE) is one of the key elements in the Question Answering task. In the past few years, there was a growing interest in NE analysis. Most current techniques for NE recognition are based on handcrafted finite state patterns [Appelt *et al.*, 1995; Weischedel, 1995], on Hidden Markov Model [Bikel *et al.*, 1999] or on Maximum entropy approach [Borthwick, 1999]

The NE analysis approach used in this task is based on a cascade of transducers. Some special features have been added to enhance the NE recognition. Among those features, a *normalization function* for normalizing proper noun occurrences in a text frame has been engineered, as well as a trivial *pronominal anaphora resolution* module. All these aspects are described further on.

For each type of NE, a transducer has been manually developed using a test corpus for validation. The transducer vocabulary is not only based on lexical information, but on semantic information too. For the purpose of NE analysis, we built several resources: list of words for entities like FIRST NAME, PROFESSION, CURRENCY and thesaurus for GEOGRAPHY for instance. Most of the expected answer types (presented in appendix 10.1) are NE recognized by our system, except for NPP entity (person names) hyponyms.

With regard to the output, XML has been used to represent the tagged documents, as shown below:

> *"<NPP>Brown</NPP>, director of the <ORGAN><CITY>Los Angeles</CITY> Centers for Alcohol and Drug Abuse</ORGAN>."*

One can observe from the previous example that embedded entities are allowed.

[1] Sinequa S.A.S.: http://www.sinequa.com
[2] LIA: http://www.lia.univ-avignon.fr/

2.1. *Normalization Function*

The identification of all the occurrences of person names is a difficult task when performed by transducers only. Many reasons could be mentioned to explain this phenomenon. The most common case is when a LAST NAME is given without any FIRST NAME. We are also aware that our resource of FIRST NAME is not (and will never be) exhaustive. This prevents us from using this semantic information in order to detect person names. However, as observed by several authors, in most cases, person names are given at least once in full form (FIRST NAME followed by eventually a MIDDLE NAME and the LAST NAME). This appears to be exact when dealing with newspaper articles (their style obeying certain editorial rules).

In order to reduce the number of unrecognized person name occurrences, a straightforward algorithm was developed based on the previous observations. First, the LAST NAME parts of the detected person name are extracted. Then, the document is parsed again in order to detect all the LAST NAME occurrences that were forgotten by the transducer. This could be done thanks to the person name previously extracted. The additional person name could then be used by the other transducers in the sequence.

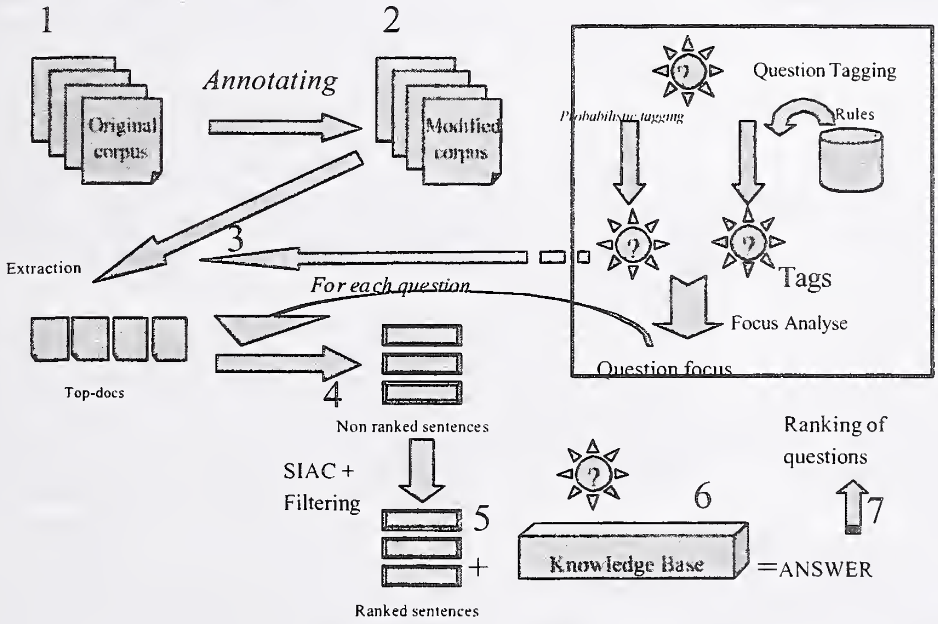


Figure 1 - From corpus to answers

In the following example, a correct normalization of the person's name "*John Paloma*" is presented:

> *"The biggest problem is identifying where these people are," said* ***John Paloma****, 36, one of the outreach workers.*
>
> ...
>
> *For that reason,* ***Paloma*** *used to stash equipment around town -- for example, high atop public toilets.*

The biggest inconvenient of this technique is that an incorrect detection of a word as a last name will affect the rest of the document processing. This mainly occurs when a first name is ambiguous (e.g. *Rose*, *France*, ...).

2.2. Pronominal Anaphora Resolution

Pronominal Anaphora is the most widespread type of anaphora. Resolving them could lead to an improvement in the Q&A task. For example, the following question expects an answer of type DATE:

> "*When did president* ***Herbert Hoover*** *die ?*"

One of the top documents found on the Internet contains the answer to that question. However, the sentence containing the answer does not contain the key element "*Hoover*", but only the anaphora "*he*":

> "*After his 1932 defeat,* ***Hoover*** *returned to private business. ...*
>
> ***He*** *died in New York City on October 20, 1964.*"

Resolving this particular case would greatly help finding the correct answer. For this reason, we have chosen to develop a Pronominal Anaphora Resolution, even though it is a "*naïve*" one. We decided to not resolve all the pronominal anaphora, but only for personal pronouns *he* and *she*, when they do not occur in quotations. The approach is based on syntactic roles of person names. A person name used as a subject is a candidate for a future anaphora resolution (according to its sex).

Although this method is quite naïve, we achieved reasonable results on our test corpus. However, we have not yet evaluated the benefit of such a resolution in the whole QA task.

3. SIAC

The SIAC information retrieval system (Figure 1 shows the Java-based GUI of SIAC) has been designed to evaluate the classification and segmentation methods we work on [Bellot & El-Bèze, 2000]. During TREC-11 Q&A track, SIAC has been used to index and to rank sentences extracted from the top-docs documents by employing some classical methods: vector space model, cosine similarity and TFIDF weighting scheme.

Let Q be a question and S be a sentence. Let u be a lemma[3], N(u) be the number of sentences containing u in the set of top-docs related to question Q, TF(u) be the frequency of u and N be the total number of sentences extracted from top-docs. The similarity between Q and S is estimated by the cosine measure (formula 1):

$$\text{cosine}(S,Q)=\frac{\sum_{u\in S\cap Q} w_{u,S}.w_{u,Q}}{\sqrt{\sum_{u\in S} {w_{u,S}}^2.\sum_{u\in Q} {w_{u,Q}}^2}} \qquad (1)$$

with:

for document words: $$w_{u,S}=TF(u,S).\left(1-\log_2\frac{N(u)}{N}\right) \qquad (2)$$

for query words: $$w_{u,Q}=TF(u,Q).\left(1-\log_2\frac{N(u)}{N}\right) \qquad (3)$$

4. Question Tagging

We defined a hierarchical set of tags corresponding to the types of expected answers (see appendix 10-1). This set was built according to a manual analysis of the TREC-9 and TREC-10 Q&A questions.

For tagging TREC-11 Q&A questions, we have developed a rule-based tagger and we have employed a probabilistic tagger based on supervised decision trees [Béchet *et al.*, 2000] for the question patterns that did not correspond to any rule.

4.1. Rule-based tagger

Our rule-based tagger is a set of Perl scripts. The main input consists on an XML file that contains 156 manually built regular expressions. These regular expressions are not exhaustive since they are based on TREC-9 and TREC-10 questions only. The following is an extract of this file: the <CITY> tag defines 3 question patterns for which the expected answer is a city.

```
<CITY>
        <s> ZTRM <\/s> (In IN in )?(([Ww]hat WP
[Ww]hat)|([Ww]hich WDT [Ww]hich)) (\w+ JJ\w? \w+
)?((city NN city)|(seaport NN seaport)|(capital
NN capital)|(town NN town))
        <s> ZTRM <\/s> What WP What is VBZ be the
DT the (\w+ JJ\w? \w+)? ((city NN city)|(seaport
NN seaport)|(capital NN capital)|(town NN town))
        <s> ZTRM <\/s> Name VB name the DT the
(\w+ JJS \w+ )?city NN city
</CITY>
```

Among the 500 TREC-11 questions, 277 questions were tagged with the rule-based tool and 223 using decision trees.

4.2. Probabilistic Tagger

The probabilistic tagger is based on the named-entity recognizer presented during ACL-2000 [Béchet *et al.*, 2000]. This recognizer uses a supervised learning method to select their most distinctive features automatically select from a set of noun phrases, embedding named entities of different semantic classes,. The result of the learning process is a semantic classification tree (a particular decision tree introduced by [Kuhn & De Mori, 1996] to classify new strings from a corpus of tagged strings) that tags an unknown entity relying on its context. The adaptation of this recognizer to this task was realized by Fréderic Béchet: the tags are not linked to a particular entity but to the question as a whole.

To "grow" decision trees, one needs a sample corpus (manually tagged TREC-10 questions in our case) and a set of key features to split tree nodes. The list of features is generated from the training corpus. Each feature corresponds to a sequence of words and/or POS tags. Splitting is made by asking whether a selected feature matches a certain regular expression involving words, POS and gaps occurring in the TREC-11 question.

In order to evaluate our probabilistic tagger, we have subdivided the 500 TREC-10 questions into two sets: a learning set (259 questions) and a test set (150 questions). Over this 150 questions test set, we obtained a 68.5% precision level for 127 questions (23 questions were not tagged because the probability of the chosen tag was less than a minimal threshold).

For example, CITY is the tag chosen for question 1204 whereas all other candidate tags have a zero probability.

[3] We used the TreeTagger [Schmid, 1994, 1995] in order to obtain POS-tags and lemmas.

Question 1204:

```
sample_1204 <s> ZTRM </s> What WP What is VBZ be
the DT the cap=tal NN capital of IN of <UNK> NP
<UNK> ? ZTRM ? </s> ZTRM=</s> = CITY


sample_1204 ACTOR_ACTRESS 0 BIOGRAPHY 0 BIRD 0
BODY_PART 0 CITY=1 COMMON_WORD 0 COMPANY 0
CONTINENT 0 COUNTRY 0 COUNTY 0<=R> CURRENCY 0
DATE 0 DEFINITION 0 DEPTH 0 DIAMETER 0 DISTANCE 0
DURATION 0 EVENT 0 EXPANDED_ACRONYM 0 EXPLANATION
0 EXPLORATOR_RESEARCHER 0 FAMOUS_NPP 0
FAMOUS_PLACE 0 FAMOUS_PLACES 0 F=OWER 0 FOOD 0
HEIGHT 0 HEMISPHERE 0 INVENTOR 0 LENGTH 0 <=R>
MEDIA 0 MINERAL 0 MONEY 0 MOUNTAIN 0 MUSICIAN 0
NUMBER 0 =THER_NP 0 PERCENTAGE 0 PHRASE 0 PLANET
0 POLITICIAN 0 POPULATION 0 RIVER 0 SEA 0 SEASON
0 SPEED 0 SPORTSMAN 0 STAR 0=00 STATE 0 TEAM 0
TEMPERATURE 0 UNIV 0 VEGETAL 0 WEIGHT 0.0<=R> 0
WRITER 0 YEAR 0
```

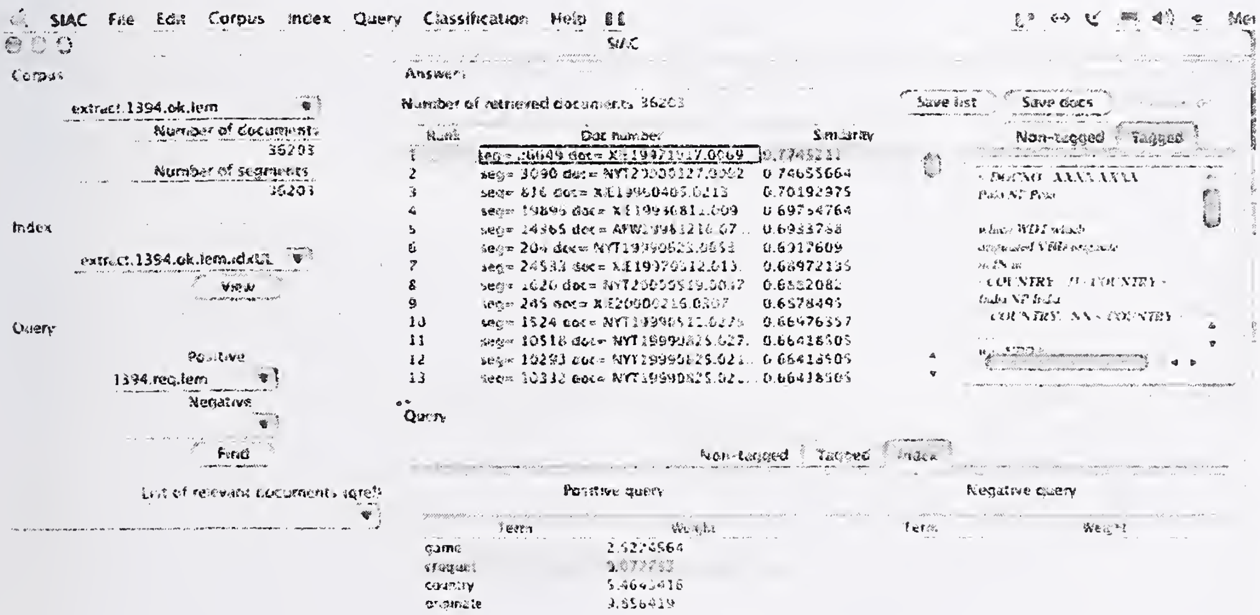
In order to tag TREC-11 questions that were not tagged by our rule-based tagger, the learning was realized over the whole set of TREC-10 questions.

4.3. *Filtering and Answer Extraction*

The sentences allowing to answer questions do not necessarily contain a word of the questions. At the opposite, a sentence may contain some keywords of the question without being related to it. Thus, a classical retrieval scheme such as similarity computation in the vector space model is not sufficient. In our case, the sentences from top-docs (the list of top-docs is the one given by NIST) are ranked by SIAC according to the similarity between them and the question. We had no time to implement a specific module to detect the focus of questions or to analyze their domain-dependent semantic properties. In order to filter sentences that probably did not contain the answer, we only kept those with a proper name appearing in the question[4] and those containing an entity of the same type than the expected answer type. This strategy prevents us from answering some questions (a NIL answer is given by the Q&A system because of the lack of proper names in the ranked sentences and/or in the question) but it enables us to select some answers more easily.

5. The Use of Knowledge Bases

We have chosen to take benefit from a set of knowledge DataBases (KDB) for several reasons, mainly: *i.)* Assess the reliability of our search engine, *ii.)* For a given relation between two NE, provide a bootstrap that may be used in the later steps of an iterative process (we plan to develop it soon). This process will be useful to extract other instances of such relations from full text collections. Therefore, it may be misleading to consider that the underlying idea of this component was to constitute a large Data Base of FAQ (Frequently Asked Questions), even though it has also been used as such.



extract.1394.ok.lem
Number of documents 36203
Number of segments 36203

extract.1394.ok.lem.idxUL

Positive 1394.req.lem
Negative

List of relevant documents (qrels)

Number of retrieved documents 36203

Rank	Doc number	Similarity
1	seg= 26649 doc= XIE19971117.0069	0.7745211
2	seg= 3090 doc= NYT20000127.0062	0.74655664
3	seg= 616 doc= XIE19960405.0213	0.70192375
4	seg= 19895 doc= XIE19940811.0009	0.69754764
5	seg= 24365 doc= APW19981216.07...	0.6953758
6	seg= 204 doc= NYT19990623.0653	0.6917609
7	seg= 24533 doc= XIE19970512.013.	0.68972135
8	seg= 1620 doc= NYT20000519.0047	0.6852082
9	seg= 245 doc= XIE20000216.0207	0.6578495
10	seg= 1524 doc= NYT19990512.0275	0.66476357
11	seg= 10518 doc= NYT19990825.027.	0.66418505
12	seg= 10293 doc= NYT19990825.021...	0.66418505
13	seg= 10332 doc= NYT19990825.02...	0.66418505

Query

Positive query

Term	Weight
game	2.5226564
croquet	3.072783
country	5.4645416
originate	3.556419

Negative query

Term	Weight

Figure 2 - The SIAC user interface

5.1. Coupling SIAC and the use of KDB

The link between a question and the production of the KDB component may be seen as a relation more than a function since the output may be multiple. To handle this (1-n) generation, we found it convenient to code the set of candidate answers using a regular expression. This regular expression is then applied on the sentences extracted by the search engine for 2 purposes: *i.)* Select the most likely answer *ii.)* Provide a support to the answer as required by the QA TREC protocol.

5.2. Some characteristics of the KDB used

5.2.1. USA topics

As it appears obviously from a quick analysis of the Q set (TREC-8 through TREC-11), several questions are focused on various attributes related to the United States of America. Thus, we have searched the net (mainly from the following url: http://www.50states.com/) in order to collect as many data related to these topics as possible. The coverage of such a "USA-centered" KDB is shown in Table 5.1.

TREC	8	9	10	11	Total
Motto	0+0	0+0	1+0	1+0	2+ 0
Flower	0+0	0+0	2+1	0+0	2+1
Song	0+0	0+0	0+0	1+0	1+ 0
Tree	0+0	1+0	0+0	0+0	1+ 0
Bird	0+0	1+0	3+0	1+0	5+ 0
Governor	0+0	0+0	1+0	3+0	4+ 0
Creation	0+0	1+0	3+0	2+0	6+ 0
Capital	1+5	2+1	0+5	1+6	4+17
Population	0+4	4+5	1+4	1+3	6+16
President	1+1	2+1	4+0	5+0	12+ 2
Total	2+10	11+7	15+10	15+9	43+36

Table 5.1. : Coverage of some KDB on the Q sets #1 centered on the US + #2 not centered on the US

It was also an opportunity to cope with similar questions when they can be asked on other countries. In each cell of Table 5.1, the first number concerns US centered questions, the second onc, other countries.

5.2.2. Book topics

In another direction, we have included in this process the relation book/author (*who wrote the book "title"?*). We have extracted from the web a list of bibliographical references. There are currently 15 800 entries in this specific KDB. Most of them come from the Pennsylvania University library and may be found at the following url: http://onlinebooks.library.upenn.edu/titles.html. We have also exploited shorter lists as the ones available at the url: http://www.state.nh.us/nhsl/bookbag/a.html.

TREC	8	9	10	11	Total
Book author	2 / 3	5 / 7	1 / 1	0 / 2	8 / 13

Table 5.2 : Coverage of the author KDB on the Q sets #1 answers produced by KDB / #2 questions on this topic

The formulation of a question is not always as precise as *who wrote the book "y"?*. Elliptic sentences as *who wrote "y"?* or *who is the author of "y"?* are more ambiguous. For instance, in Q8/196, *"Hamlet"* may be a movie or the famous play. The case is also encountered in Q11/1759: *"Fiddler on the Roof"* may be a novel or a musical. The novel was not in our KDB and it is a chance since only the musical has been considered as the correct answer by the judges. Whether we decide to enrich our resource or not, we have to take this kind of difficulty into account.

5.2.3. Archives

It was also natural to check whether questions found in TREC-11 were not already present in previous TRECs. In such case, the answer provided could be reused. Let us call an Archive A_i, a pair of two sets: (Questions, Answers) of $TREC_i$. Until we got A_{11} (the patterns of $TREC_{11}$), we have considered the following: for Q_8 use A_{9+10}, for Q_9 use A_{8+10}, for Q_{10} use A_{8+9}, for Q_{11} use A_{8+9+10} (1st line of Table 5.3). As shown in 2nd line of Table 5.3, the coverage on Q_{8-10} does not increase a lot when A_{11} is also taken into account, except for Q_{10}.

TREC	8	9	10	11	Total
# Q	0	4	5	5	14
Including A_{11}	0	4	9	5	18

Table 5.3: Considering other Q sets as FAQ

Note that we did not search for a similar question but for exactly the same one. Therefore, some improvements can be made here.

5.2.4. Typos and Variants

Typos may be seen as a noise disturbing the canal between the input (Q) and the output (A). For a question such as Q10/1249/ *Who wrote "The Devine Comedy"?* the relation (Dante – Divine Comedy) included in the KDB described in 5.2.2 could not be exploited. We have used thc classical edit distance [Lowrance & Wagner, 1975] and the dynamic time wrapping method to find the optimal way to associate words as Divine and Devine. Penalty weights havc been assigned to operations (substitution, omission, insertion), and a threshold has been empirically chosen in order to avoid confusion such as *Mexico/Monaco*. This procedure is not only useful to handle typos but also to cope with the numerous variants, which can be observed for the Proper Nouns transcription of Foreign Entities (there are, for example, more than 50 ways to write *Kahdafi*. This can be coded by a regular expression *[GK]h?ah?dd?h?ah?ff?i*).

[4] The proper name detection was realized according to the POS-tags.

As far as we want to take into account human factors, we have chosen to generate an answer where the graphemes involved are the most similar ones and not necessarily the ones used in the question. Our assumption is that the user will find more acceptable a system answering sometimes to another question than a system giving a wrong answer to his question.

5.3. KDB Summary

In the subsections 5.2.1 to 5.2.3, we have given some examples of the domains covered by the KDB we used. They correspond to about half of the answers currently supported by our KDB component. The second half concerns various topics such as rivers, mountains, Nobel's, hurricanes and so on. It is impossible to describe each of them in detail here, but it is interesting to see that the coverage is more or less the same on each TREC.

TREC	8	9	10	11	Total
Answer KDB	22	73	64	61	220
# Questions	200	694	500	500	1 894
%Q handled	11.0	10.5	12.8	12.2	11.6

Table 5.4: Global Coverage of 36 KDB

While for 12.2 % of the Q_1 set, the KDB are able to produce an answer, it is not possible to insert all of them in our run. As mentioned in section 5.1, we have also to match each answer with the output of SIAC. Sometimes (8 / 61 cases) the search engine is too silent, therefore the set of candidates may be empty. In more than half of the cases (35 / 61), it was possible to find a pattern matching the regular expression. For the remainder (18 / 61 cases), no match has been found in the sentences retrieved by SIAC.

6. Ordering answers

This year's QA track introduced newness in the evaluation measure in such a way that systems have to cope with the following principle: rank the answers from the most reliable to the less one. In order to take into account this requirement, our answers have been ordered according to results provided by the use (or non-use) of knowledge databases (KDB) – as a way to validate an answer – and by the question classifier output. So, for each question, the question classifier assigns one (or several) expected NE(s) and its (their) corresponding confidence(s). If it cannot be decided which of the 44 available entities should be responsive, the question is tagged as "unknown". From these points, our ordering strategy can be summarized schematically as follows: divide the Q-set in three main groups,

- Q1: questions for which answers have been found by SIAC and validated with KDB. Since there is an agreement between two independent components, it is justified to assign a highest reliability score to the group produced by such a combination and to place it at the top ranks. Thirty-five questions were in this group and were ranked from 1 to 35.
- Q2: questions for which answers have been found only by SIAC and not covered by any database. This group, the major one with 438 questions, could be divided in two parts: non NIL answers (389) and NIL answers (49). As described in section 4.3, filters are applied on SIAC output in order to keep only expected entities mapping question class(es) – it may happen that all the candidates are eliminated by this filtering – that is how NIL is produced by the system. It was decided to put these NIL at the end of this group, as they are the results of many treatments and therefore the decision process becomes too uncertain. Inside non NIL answers, order was defined first by decreasing confidences (in question classes) and second by question classes. Order among question classes (see table 6.1) has been derived from previous experiments performed for tuning purpose. For example, our classification component performs well for questions asking for YEAR and DATE, and as named entities mapping these classes are also well detected, we are more confident in answers coming from these series. On the other hand, by the time of our participation, for questions asking for frequencies, named entities finder was not able to detect these expressions – accordingly it should be risky to bet on the class mapping this entity.
- Q3: questions for which the classifier did not assign a class (and tagged "unknown"). This is clearly a flaw in our system's answering process as answer selection depends on these classes. Therefore, such questions will be answered with a NIL and put at the bottom ranks. It happened thirty times over the entire TREC-11's set but three of them were finally over-handled by KDB – and backed up in the first group Q1. The remainders (27) have been left as NIL.

This ordered list (Q1, Q2, Q3) corresponds to the way we ranked the three groups.

YEAR, DATE, COUNTRY, COUNTY, NPP, ACRONYM, CITY, MAIL, MONTH, URL, STATE, ADDRESS, TITLE, LOCATION, ORGAN

Table 6.1: Top 15 questions classes (ordered by preference)

7. Experiments and results

7.1. *Official results*

Table 7.1 shows the results obtained by our run LIA2002a (only one run was submitted).

Number wrong (W):	440
Number unsupported (U):	4
Number inexact (X):	4
Number right (R):	**52**
Confidence-weighted score (CWS):	**0.246**
Precision of recognizing no answer	7 / 75 = 0.093
Recall of recognizing no answer	7 / 46 = 0.152

Table 7.1 : Official results

7.2. *Experiments*

After the dead line, we performed some additional experiments. It was possible to evaluate them thanks to the TREC-11 answers patterns made available by Ken Litkowski. For this purpose, a home-made tool was developed to compute the confidence weighted score ("CWS"). In the following, these new experiments will be referred as LIA2002o (o standing for October) and LIA2002n (n for November).

- **Evaluation of the KDB contribution:**

The results reported in table 7.2 are useful to focus only on the behavior of questions for which a KDB was involved. For this, we assume that the other answers (from rank+1 to 500) were wrong:

	#	R	U	CWS
Lia2002a	30	24	4	0.051
Errors at rank	6, 11U, 15, 16U, 21U, and 29			
Lia2002o	33	26	5	0.056
Errors at rank	6, 11U, 16, 17U, 22U, 30, and 33U			
Lia2002n	35	28	5	0.061
Errors at rank	7, 15, 17U, 19U, 24U, 32, and 35U			

Table 7.2 : KDB Contribution

Where:

- "R", "U", "CWS" stand respectively for "right", "unsupported", and "confidence weighted score".
- "Lia2002a" is the run submitted in August for TREC-11,
- "Lia2002o" is a run with few additions in KDB. Also, minor bug corrections inside our whole system and specially in ordering strategy were done (ordering for SIAC answers was broken in our TREC submission)
- "Lia2002n" is our last run. It includes two more entries (a tiny extension of the KDB). The main difference with the previous ones is that answers powered by KDB are ranked by applying the same ordering strategy as answers from SIAC.

- **Answers allocation** (table 7.3): System succeeded in finding a non nil, right and supported answer in about 10% of the cases (column reported as "R-nil"). It provided document containing a correct answer in 15% of the cases (column reported as "D") but failed to extract it in about 5% of the cases (column reported as "D-(R-nil)"). SIAC was able to find 5% of the non nil correct answers but ten of them were overlapped by the KDB.

#	KDB Size	**R**	**U**	**R-nil**	**D**	**D-(R-nil)**
Lia2002a	30	53	5	**45**	72	27
Lia2002o	33	55	7	**47**	74	27
Lia2002n	35	55	7	**47**	74	27
Without any KDB	0	34	37	**26**	59	33

Table 7.3 : Answers Allocation

- **Evaluating ordering strategy:** Table 7.4 presents CWS results obtained only by correcting our ordering strategy as we intended it to be. It provided a gain of about 7% just by re-ordering 55 answers.

	R	CWS (1)	CWS (2)	**Gain %**
Lia2002a	53	**0,246**	**0,268**	+ 9%
Lia2002o	55	**0,258**	**0,278**	+ 8%
Lia2002n	55	**0,266**	**0,285**	+ 7%
Without any KDB	34	**0,084**	**0,139**	+ 65%

Table 7.4 : Ordering Strategy Gain

(1) Answers ordered as for August submission,
(2) Answers ordered by using planned ordering strategy.

8. Conclusion

For our first participation in TREC - question answering, we focused on a small number of questions, that is questions for which an answer can be produced with a sufficient level of confidence. The goal was to reach 30% of accuracy which is honorable as a first trial.

A lot of work remains. Firstly, we could have gone into entity recognition in greater depth, using more statistics. Secondly, because two different tools have been used in order to tag (NE) the documents and the questions, we experienced some problems making a mapping from one to the other. The lack of compatibility should be solved by using the same set of tag. Also, anaphora resolution is too simple and could be applied on many other anaphora phenomena. Another important point: question tagging is quite weak. For example, for many questions, it assigns the same confidence to different tags. The selection of the tag to be considered could be easily improved. Moreover, answer extraction is too much simple. Because no syntactic tagging is done, it is impossible to choose precisely a phrase in which the answer is supposed to be. So, the only thing we did was to extract the searched entity wherever it was in the candidate sentence. Consequently, many wrong answers were retrieved.

Relying on some knowledge bases clearly improves the results of our system. Typo correction is quite efficient and allows us to answer correctly several questions. We can improve the cases where an answer is provided by the KDB and SIAC fails to retrieve any expected NE, by enriching the question with this answer in order to retrieve supporting documents. Moreover, we could increase the coverage of this KDB in two directions: i.) Find other knowledge sources on more and more subjects, ii.) Use each KDB as a bootstrap in order to enlarge it thru text extraction. We consider that the second item is a key point to make the first one feasible.

Finally, let us consider the graph plotted in figure 3. It represents the growth of correct answers. We can see that the curve grows in stages. Important improvements are followed by long flat lines.

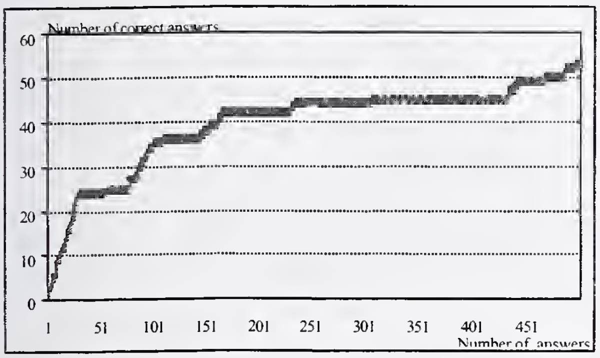


Figure 3: Number of correct answers by number of answers

In fact, there are 5 big stages:

Stage	Correct answers	Accuracy	Length of stage
1-30	24	0.8	30
78-110	11	0.33	33
145-166	6	0.27	22
433-443	4	0.36	11
484-495	3	0.25	12

Table 8.1: Location of correct answers in the list

If we do not consider the first stage (due to the KDB), we have 4 stages (which length is between 11 and 33) where accuracy is quite good (between 0.25 and 0.36). This concentration is sufficiently significant to conclude that some questions have the same behavior and the system performs quite well on these types of questions. Since they are grouped, it should be possible to detect and locate them higher in the list. It could be possible to improve the results by detecting these types of questions.

This concerns 48 questions, that is more than 90% of our correct answers. If they were located at the beginning of the list, the CWS would be 0.32 instead of 0.246.

9. References

[Appelt *et al.*, 1995] D. E. Appelt, J. R. Hobbs, J. Bear, D. Israel, M. Kameyama, A. Kehler, D. Martin, K. Myers, & M. Tyson, "*SRI International FASTUS System MUC-6 Test Results and Analysis*", Proceedings of the Sixth Message Understanding Conference, Columbia, Maryland: Morgan Kaufmann Publishers, pp. 237–248, 1995.

[Béchet *et al.*, 2000] F. Béchet, A. Nasr, F. Genet, "*Tagging Unknown Proper Names Using Decision Trees*", in Proc. of ACL'2000, Hong-Kong, China, pp.77-84, 2000.

[Bellot & El-Bèze, 2000] P. Bellot, M. El-Bèze, "*Clustering by means of decision trees without learning or hierarchical and K-Means like algorithms*", in Proceedings of RIAO'2000, Paris, France, pp. 344-363, 2000.

[Bikel *et al.*, 1999] D. M. Bikel, R. Schwartz and R. M. Weischedel, "*An Algorithm that Learns What's in a Name*", Machine Learning, Special Issue on NLP, 1999.

[Borthwick, 1999] A. Borthwick, "A Maximum Entropy Approach to Named Entity Recognition", Ph.D. (1999) New York University. Department of Computer Science, Courant Institute. Specialized in artificial intelligence and computational linguistics.

[Kuhn & De Mori, 1996] R. Kuhn, Rde Mori, '*The application of semantic decision trees to natural language understanding*", IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(5), pp. 449-460, 1996.

[Lowrance & Wagner, 1975] R. Lowrance & R. A. Wagner, "*An extension of string-to-string correction proble.*", Journal of the ACM, 22(2), 177-183, 1975.

[Schmid, 1994] H. Schmid, "*Probabilistic Part-of-Speech Tagging Using Decision Trees*", in Proc. of the First International Conference on New Methods in Natural Language Processing (NemLap-94), Manchester, England, pp. 44-49, 1994.

[Schmid, 1995] H. Schmid, "*Improvements In Part-of-Speech Tagging With an Application to German*", EACL SIGDAT Workshop, Dublin, Ireland, In: Feldweg and Hinrichs, eds., Lexikon und Text, pp. 47-50, 1995.

[Weischedel, 1995] R. Weischedel, "*BBN: Description of the PLUM System as Used for MUC-6*", Proceedings of the

Sixth Message Understanding Conference, Columbia, Maryland: Morgan Kaufmann Publishers, pp. 55-69, 1995.

10. Appendix

10.1. Hierarchical List of Expected Answer Types

```
MISC
        ACRONYM
        EXPANDED_ACRONYM
        ADDRESS
        ZIP
        PHONE
        URL
        EMAIL
        AGE
        DIMENSION
                ELEVATION
                WIDTH
                DIAMETER
                HEIGHT
                DEPTH
                AREA
                VOLUME
        PHRASE
                DEFINITION
                EXPLANATION
                EVENT
                TITLE
                        BOOK
                        FILM
                        MUSIC
                        PAINTING
                BIOGRAPHY
                COMMON_WORD
                        CLOTHES
                        VERB
                        TOY
        PROFESION
        OTHER_NUMERAL
                DISTANCE
                MONEY
                NUMBER
                ORDINAL
                PERCENTAGE
                        CONVERSION_RATE
                POPULATION
                QUANTITY
                SPEED
                TEMPERATURE
                WEIGHT
        TIME
                DURATION
                FREQUENCY
                DATE
                        BIRTHDAY
                        WDAY
                        DAY
                        MONTH
                        YEAR
                        HOUR
NP
        ANIMAL
                BIRD
                INSECT
        BODY_PART
        COLOR
        CURRENCY
        DISEASE
        FIRSTNAME
        FOOD
        LANGUAGE
        MINERAL
        MUSICAL_INSTRUMENT
        NICKNAME
        SPORT
        VEGETAL
                FLOWER
                FRUIT
                VEGETABLE
        PROPER_NOUN
                ACTOR_ACTRESS
                CHAIRMAN
                FAMOUS_PERSON
                        NOBEL_PRIZE
                INVENTOR
                MUSICIAN
                PAINTER
                POLITICIAN
                        PRESIDENT_POLITICIAN
                SPORTSMAN
                EXPLORATOR_RESEARCHER
                        EXPLORATOR
                        RESEARCHER
                WRITER
        LOCATION
                CITY
                        CAPITAL
                CONTINENT
                COUNTY
                COUNTRY
                FAMOUS_PLACE
                HEMISPHERE
                LAKE
                MOUNTAIN
                PLANET
                RIVER
                SEA
                STARS
                STATE
                LOCATION
        ORGANIZATION
                COMPANY
                        STORE
                UNIVERSITY
                MEDIA
                        JOURNAL
                        TV
                        RADIO
                TEAM
        OTHER_NP
        SEASON
UNKNOWN
YES_NO
```

The question answering system QALC at LIMSI: experiments in using Web and WordNet

G. de Chalendar, T. Dalmas, F. Elkateb-Gara, O. Ferret, B. Grau, M. Hurault-Plantet,
G. Illouz, L. Monceaux, I. Robba, A. Vilnat
LIMSI – CNRS (France)

1. Introduction

The QALC question answering system at LIMSI (Ferret et al, 2001) has been largely modified for the TREC11 evaluation campaign. Architecture now includes the processing of answers retrieved from Web searching, and a number of already existing modules has been re-handled. Indeed, introducing the Web as additional resource with regard to the TREC corpus, brought us to experiment comparison strategies between answers extracted from different corpora. These strategies now make up the final answer selection module.

The answer extraction module now takes advantage of using the WordNet semantic data base, whenever the expected answer type is not a named entity. As a result, we draw up a new analysis for these question categories, just as a new formulation of associated answer extraction patterns. We also changed the weighting system of the sentences which are candidate for answer, in order to increase answer reliability. Furthermore, the number of selected sentences is no longer decided before extraction module but inside it according whether the expected answer type is a named entity or not. In the last case, the number of selected sentences is greater than in case of a named entity answer type, so as to take better advantage of the selection made by means of extraction patterns and WordNet.

Other modules have been modified: the QALC system now uses a search engine and document selection has been improved through document cutting into paragraphs and selection robustness improvement. Furthermore, named entity recognition module has been significantly modified in order to recognize precisely more of them and decrease ambiguity cases.

In this paper, we first present the architecture of the system. Then, we will describe the modified modules, i.e. question analysis, document selection, named entity recognition, sentence weighting and answer extraction. Afterwards, the strategies of final answer selection of the new module will be described. Finally, we will present our results, ending with some concluding remarks.

2. Architecture

QALC system core is made of classical following modules: question analysis, document selection, named entity recognition and answer extraction. As input of the system, we use the same TREC11 question set, but two different corpora, on the one hand the TREC11 corpus and on the other hand the Web. Figure 1 shows the architecture of the system. In this figure, various arrows indicates TREC11 corpus processing and Web documents processing. Selected answers provided by the two processing chains are merged in the final answer selection module (see section 4).

The Web chain of QALC is nearly the same as the classical one, except that the answers are looked up in documents gathered from the Web instead of in the QA Track corpus. The idea behind that is, as in (Soubbotin and Soubbotin, 2001), that there is a great chance to find the answer to a question in the Web in a shape similar to the one of the question itself, but in an affirmative form. This hypothesis is based on the huge quantity of documents directly available on the Web and thus on their high redundancy. So, for the question "When was Wendy's founded?", we expect to be able to find a document containing the answer in the form: "Wendy's was founded on ...". We want to search the Web for strings with exact match. In the previous example, we don't want to search for documents containing one of the words of the query. Rather than that, we want to search for the exact phrases. For such a goal, a boolean search engine able to accept queries made of multiple words terms is necessary. Google is such a tool.

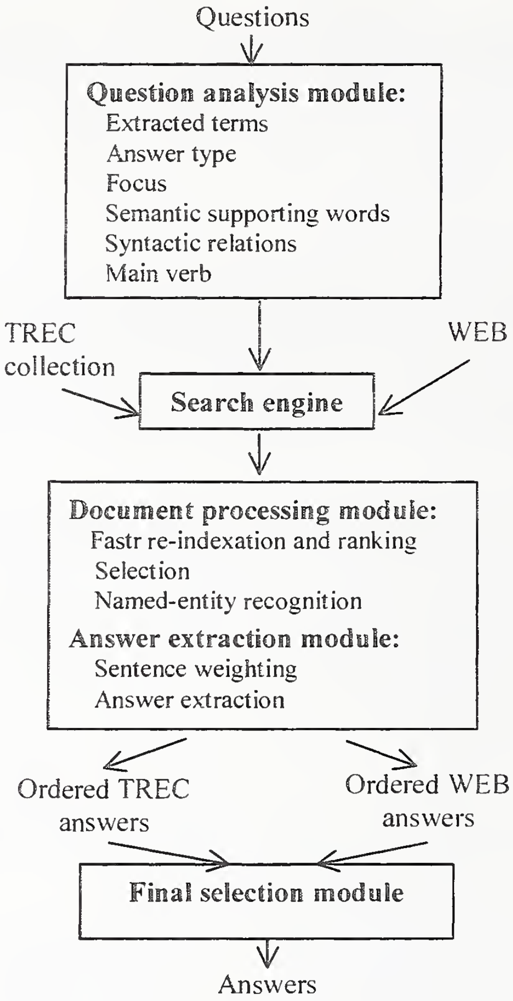


Figure 1. QALC System Architecture

The most important part for this chain is the rewriting of the question. It uses the results of the questions analysis and rewriting rules manually written from the study of the TREC 9 and 10 questions. The questions are categorized in function of their type and their category. If these two indices are not sufficient to adequately rewrite a question, a lexical marker is used. For example, the question "What continent is Egypt on?" has for type LOCATION-STATE and is of the category WhatGNbeGN. The focus of this question is "Egypt" and the kind of location (typeGen) to find is a "continent". The rule used to rewrite this question is:

(type = LOCATION-STATE) & (category = WhatGNbeGN) & (lexical-modifier = on)

=> "<focus> <verb> on" <typeGen>

So, the question will be rewritten in: ""Egypt is on" continent".

Here, the quotes are important, they indicate to Google to search for documents containing this exact string. This query finds 50 answers on Google and the first one contains the string: "Even though Egypt is on the continent of Africa...".

We try to make the rules as specific as possible in order that the documents gathered by the queries they generate have the highest probability to contain an answer to the question. The drawback of this approach is that the probability not to find any document corresponding to the query increases with the specificity of the rules. To counter this tendency, we systematically propose several rules for each question type ordered by descending specificity. The most general one is simply the question without any edition.

The remaining of the Web chain is exactly the same as the classical one. At the end of the chain, we have one or more propositions of solutions. We cannot submit them directly as answers in the TREC QA track as they are not supported by documents of the TREC11 corpus. So, they are used to do a second path in the chain, this time querying the QA corpus with our search engine with queries enriched by the Web candidate answers. The answers that are found in the Web and that are confirmed by the corpus see their score greatly increased compared to the answers found only in the corpus.

In the future, we plan to use the work done with the Web chain more completely by using the rewritten questions to search the TREC corpus. This would improve the results of the search engine for the documents containing answers expressed similarly to the question. Such question-document pairs are certainly less numerous than in the Web but there are certainly a certain amount of them and we should not miss them.

3. QALC main processing chain

3.1 Question analysis

Question analysis is performed with the aim of extracting from the question the essential information required for retrieving an answer within a document sentence. When the expected answer type is a named entity, then essential information is the named entity type itself. In this case, analysis is based on the recognition of lexical and syntactical clues within the question that determine the named entity type of the answer. This year, we improved our system by analysing more precisely some named entity types, in particular physical numerical entities, and by adding some units to those already used.

But, if the expected answer is not a named entity, then the essential information are answer extraction pattern, and semantic relations between the answer and words of the question. Thus, the goals of the analysis are first to determine the question type in order to associate with it an extraction pattern, and then to build a validation schema that will instantiates the parameters of the pattern. The validation schema is built from the semantic supporting words of the question, i.e. the words that will have semantic relations with the answer within the sentence. For this purpose, a local syntactical analysis first locates within the question the grammatical features of a same question type. This analysis allows the system to recognize the question type and the semantic supporting words that will be used to produce the validation schema. Then, the validation schema is built from a set of production rules that uses the analysis results. Let us consider for example the question 1505:

Question : What is the currency used in China?
The analysis of the question gave the following features structure:
Question qMoney :
 Answer extraction pattern: aMoney
 Semantic supporting words:
  Answer type: currency
  Focus: China

From this analysis results, the production rules then gave the following validation schema:

Hypernym: monetary unit
Country: China

This schema will be subsequently used by the extraction pattern named *aMoney* (see section3.5.2).

3.2 Document selection

QALC system uses two subsequent document selections, the first one performed by a search engine, and the second one performed by FASTR, adding variants of the question words as new criterion. For the first selection, we used MG[1], a boolean search engine. Our choice was to cut out the collection documents in paragraphs of approximately the same size. By this way, we add to the selection criteria a criterion about their proximity. The query was built from the question analysis results, by combining lemmatized and non lemmatized forms of the content words and adding the focus terms in order to reinforce this selection criterion.

Concerning the second selection, we made some improvement. The way the documents that are likely to contain an answer to a question are selected from the results of the search engine is globally the same in this version of QALC as in its previous versions. First, a set of one-word and multi-word terms is extracted from the question by a pattern-based termer. Then, the documents retrieved by the search engine are indexed by the FASTR tool (Jacquemin, 2001), which recognizes in the documents the terms extracted from the question, as well as their morphological, syntactic or semantic variants. Each recognized term is weighted according to the kind of variant it is and a score is computed for each document by aggregating the weights of its indexes. Finally, a restricted set of documents is selected when a significant shift in the documents' scores is found. Otherwise, a fixed number of them, in this case 100, is taken.

For the TREC11 evaluation, we specifically took into account a problem we had noticed in the previous evaluations. FASTR was designed for recognizing complex variants of terms in the field of terminology but not as a robust tool for information retrieval. For instance, the errors of the morpho-syntactic tagger we use, in this case the TreeTagger tool (Schmid, 1999), have a significant impact on its results. As a consequence, FASTR misses some non-variant occurrences of questions' terms that can be recognized by a basic term matcher. More precisely, we found that for one-word and multi-word terms without variation the recall of FASTR is equal to 71%. This evaluation was done with the documents selected for the 500 questions of the TREC11 evaluation. The reference was set by a term matcher working from the results of TreeTagger.

	QALC TREC 10	NIST-100	QALC TREC 11
selected documents with an answer (nb)	2041	2479	2313
selected documents (nb)	30992	49900	34568
recall (%)	46.0	55.8	52.1
precision (%)	6.6	5.0	6.7
selected documents with an answer - variation (%)	reference	+ 21.5	+ 13.3
selected documents - variation (%)	reference	+ 61.0	+ 11.5

Table 1: Document selection results for TREC10 questions

In order to improve the selection of documents, the version of QALC for the TREC11 evaluation combined the results of FASTR and those of our basic term matcher: the terms found by the term matcher are added to those found by FASTR and the doubles are discarded. The impact of this change is shown in Table 1. NIST-100 corresponds to a selection module that would always take the first 100 documents returned by the search engine. The results of Table 1 are based on the list of judgments about participants'

[1] Managing Gigabytes is an open source indexing and retrieval system. Its homepage may be found at: http://www.cs.mu.oz.au/mg

answers given by NIST for the TREC10 evaluation. The precision measure is the ratio between the number of selected documents that actually contain an answer and the number of documents selected by the considered system. The recall measure is the ratio between the number of selected documents that actually contain an answer and the number of documents found by at least one participant of TREC10 and that contain an answer. Table 1 shows that the combining of two term recognizers, a basic one and a more sophisticated one, is an interesting strategy as it makes both recall and precision increase. Moreover, a significant number of new relevant documents are found while the number of documents to process by the answer extraction modules only increases linearly.

3.3 Named entity recognition

The named entity module identifies named locations, named persons, named organization, dates, times, monetary amounts, measures and percentages in text. For Trec11, we have developed a new system for locations, persons and organizations recognition. In order to identify these entities, our system uses hand-made rules in which we specify named entities structure in term of text tokens and what we can find about them from resources such as tagger, morphosyntactic analyzer and knowledge base of names, clue words and abbreviations.

The system performs in three stages:
- Preprocessing: we use treetagger to tokenize the input text and tag them with a syntactic and typographic category.
- Database lookup: for each token or group of token, we check in a list of known names, abbreviations and clues. If a token is found in a database, this information is added to token feature.
- Named Entity analyzer uses language specific context-sensitive rules based on word features recognition pattern matching. For each token, we look for the longest pattern of token features that matches with pattern rules.

Features used for tokens are:
- Lookup in knwoledge base of names (persons, organizations, locations).
- Lookup in knowledge base for names, organizations and locations clues.
- Lookup in knowledge base for firstnames abbreviations.
- Typographic features (Capitalization, Roman characters, etc).
- Syntactic category.
- Lemma.

For TREC11 we wrote:
- 7 rules for Organizations,
- 9 rules for Locations,
- 7 rules for Persons.

Here is a example of detecting location entity:

If the (syntactic category of token (or group of tokens) is "Proper noun") and these tokens are followed by a token found in State clues Database, we identify all these tokens as a location.

3.4 Document sentence weighting

All the sentences in selected documents are analysed in order to give them a weight that reflects both the possibility that the sentence contains the answer, and the possibility that the QALC system locates the answer within the sentence. The criteria that we used produce simple processing, and are closely linked with the basic information extracted from the question. The resulting sentence ranking do not have to miss obvious answers. Thus, the main goal of this analysis is to assign a higher weight to sentences which contain most of obvious information. In return, all sentences are kept for subsequent processing, except those which do not contain any word from the question. Our aim is that the answer extraction modules can raise to an upper rank a lower weighted answer thanks to added specific criteria.

The criteria that we retained use the following features retrieved within the candidate sentence:

1. question lemmas, weighted by their specificity degree,
2. variants of question words,
3. exact words of the question,
4. mutual closeness of question words,
5. word whose type is the expected answer named entity type.

The specificity degree of a lemma depends on the inverse of its relative frequency within a large corpus. The criterion of mutual closeness of question words is the closeness between them arranged in pairs in the sentence.

First we compute a basic weight of the sentence based on the presence of question words within the sentence, and then we add weights from the other criteria. The computation of the basic weight of a sentence is made from lemmas (or from words if the word is unknown for the tagger), and their specificity degree. Some words are not taken into account, i.e. determinants or prepositions, transparent nouns, and auxiliary verbs. According to Fillmore (2002), a transparent noun is a noun whose complement is semantically more relevant that the noun itself. For instance, the word *name* is transparent in the question 1396 *What is the name of the volcano that destroyed the ancient city of Pompeii?*, and *volcano* is the semantically relevant noun. We made an a priori list of such words.

Thus, the basic weight of a sentence is given by:

$$P = (dr_1 + \ldots + dr_i + \ldots + dr_m) / (dq_1 + \ldots + dr_j + \ldots + dq_n)$$

with:
dr_i: specificity degree of a lemma from the question found in the sentence,
dq_j: specificity degree of a lemma of the question,
m: number of lemmas found in the sentence,
n: number of lemmas in the question.

Each lemma is taken into account only once even if it occurs more than once in the same sentence. If a word from the question is not found in the sentence, but a variant of it, half of the specificity degree of the word is added to the basic weight of the sentence. As the basic weight is relative, its maximum is equal to one. We bring it to 1000 for convenience. We subsequently add an additional weight to this basic weight for each additional criterion that is satisfied. Each additional weight cannot be higher than about 10% of the basic weight.

3.5 Answer extraction

3.5.1 Named entity answer type

If the expected answer type is a named entity, then selected answers are these words within the sentence that correspond to the expected type. In order to extract the answer, the system first selects, among the sentences provided by the sentence weighting module, the ten sentences that have the best weight. Then, it computes additional weights taking into account:

1. exact or generic named entity type of the answer,
2. location of the potential answer with regard to the question words within the sentence,
3. redundancy of an answer.

A sentence may not contain a word corresponding to the expected named entity type, but to a more generic one, for instance NUMBER instead of DATE. In that case, a lower weight will be given to the generic type. An additional weight is then given to potential answers closest to the question words. This closeness is computed with regard to the barycentre of the question words within the sentence. If there is more than one potential answer in the sentence, the one, that will be selected, is the closest to the question words. Finally, if a same potential answer is retrieved more often than others in the ten sentences, then it is assigned with an additional weight. These criteria allow the system to rank better a potential answer that had not the best weight as sentence. For instance:

Question 1475: *Who was the first person to reach the south pole?*
Candidate sentences:

1210 NYT19981103.0190 < PERSON> Diana Preston <e_enamex> s absorbing and moving story of the attempt by the British explorer < PERSON> Robert Falcon Scott <e_enamex> to be the first to reach the South Pole shows that that reverence for the noble failure is not unique to Japan.

1185 NYT19991028.0488 The truly adventurous go all the way to the South Pole , first reached in 1911 by< PERSON> Roald Amundsen <e_enamex> , a Norwegian .

1165 NYT19991129.0264 The Norwegian < PERSON3> Roald Amundsen <e_enamex> led the first successful expedition to the South Pole , reaching it on Dec. 14 , 1911.

The weight of each sentence is indicated in the first column. The lower weight of the second sentence comes from the exact word criterion (*reached* instead of *reach*). On the other hand. the third sentence has a lower weight due to less closeness of words. After having computed the added weights. we then obtained the following answers:

1365	NYT19991028.0488	Roald Amundsen
1267	NYT19981103.0190	Robert Falcon Scott

In that case, the redundancy criteria brought the correct answer to the first rank.

3.5.2 Common noun or phrase answer type

Each candidate sentence provided by the sentence selection module is analysed using the extraction pattern determined by question analysis. This pattern uses the associated validation schema to instantiate its parameters.

Extraction patterns are composed of a set of constraint rules on the candidate sentence. Rules are made up of syntactic patterns that are used to locate potential answer within the sentence, and of semantic relations that are used to validate answer. Syntactic patterns locate connecting words and simulate possible paraphrases of the answer. Semantic constraints are proved using WordNet. Extraction patterns are implemented through automata whose transitions are a set of functional constraints that have to be satisfied by the feature structure, representing the sentence, being valued. Potential answers are weighted according to the satisfied constraints. The weight amount depends on the reliability of the constraint. For instance, the hypernym relation constraint is given a high weight as it is a reliable relation.
Let us consider the example analysed in section 3.1:

Question: What is the currency used in China?
Answer: yuan
Extracted from the sentence:
The central bank governor acknowledged that the Renminbi yuan , China s currency , is now facing pressure for further appreciation due in part to growing foreign exchange reserves which reached 126 billion US dollars at the end of July .

The retrieved answer satisfies the following constraints:

Syntactic extraction pattern:
Answer , Country <currency | money>
Semantic validations:
Monetary unit *is hypernym of* yuan
Answer *has in gloss* Country

With *Country* instantiated by *China*. according to the validation schema.

At the output of the sentence weighting module. 12 candidate sentences had the same highest weight equal to 1120. Among these sentences. 5 contained the correct answer. The answer extraction criteria. that the QALC system used. thus allowed it to select the correct answer.

4. Final answer extraction module

In TREC11 evaluation, we had to supply one answer per question and the set of 500 answers had to be ordered according to a confidence score. As explained before, this year, we elaborated two different search strategies: the main strategy searches the answers only in the TREC collection, while the Web strategy searches the answers on the Web and then tries to confirm these answers by locating them also in the TREC collection.

Moreover, these two strategies supply for each question, a set of answers (but we examine at most the first five), which are ordered according to the score they received during the sentence weighting and answer extraction processes. The role of the final selection is hence to choose a unique answer between these two sets. For this selection we worked out two different algorithms applying a sequence of rules that we briefly present here:

If the same first answer is found by the Web strategy and by the main strategy, this answer is returned with an augmented score, the score increase being more significant if the answer is not NIL.
If a first answer is found by the main strategy and if the first answer of the Web strategy is NIL, the main strategy first answer is returned with its original score (or conversely but only if the Web answer is confirmed).
If a first answer is found by the Web strategy, but if this answer is not confirmed and if the answer of the main strategy is NIL, the answer NIL is returned with its original score.

When any of the preceding rules can be applied, we tested two different algorithms; both of them take into account not only the first answers of the sets but also possibly the other answers of the two sets.

The first algorithm consists in increasing the score of the first answer of the main strategy, provided that this first answer is also present in the Web strategy results. If the first answer is not found in the Web results, the other way round, the first answer of the Web strategy is searched in the main strategy results. If the two searches fail the main strategy first answer is returned with its original score.

The second algorithm attributes a score to each couple (i,j), i being the position of an answer in the answer set of the main strategy, j being the position of an answer in the answer set of the Web strategy. The score is especially high since the two answers are equal. The answer of the couple obtaining the best score is finally returned with a score that is augmented according both positions: i and j.

Example:

Question 1806: When was the first heart transplant?
EN - Answers:
Answer 1: in 1979,
Answer 2: in 1967 ...
Web - Answers:
Answer 1: in April 1985,
Answer 2: on December 3, 1967,
Answer 3: in 1968,
Answer 4: in 1967, ...
Final – Answer: in 1967 obtained by couple (2,4)

5. Results

Table 2 presents the results we obtained in TREC11 evaluation for the three runs we submitted. As explained before, we try a new solution to select the pertinent documents, using another engine MG (run 3), with documents cut in paragraphs. We also combine FASTR with a basic term matcher to select documents (as illustrated in section 3.2). To evaluate the advantages of these strategies, we also use the results provided by NIST. So the two first runs are obtained with NIST search engine and the third one with MG. The first run considers FASTR alone, the second and third ones consider the combination of FASTR with a basic termer. The first run uses the first algorithm to choose the final answer (section 4), the two others use the second one. All of them take advantage of the results of the Web search.

	W	U	X	R	TREC11 score	NIST pattern score
Run 1 — NIST 1 & Web	342	21	7	130	0.485	0.567
Run 2 — NIST 2 & Web	336	20	11	133	0.497	0.587
Run 3 — MG & Web	330	20	11	139	0.488	0.572

Table 2: Results of QALC system three runs

A first conclusion is that we obtained this year more than a quarter of right answers which represents a quite better score than last year. Even when considering that the question set of this year did not contain complex questions as the definition questions of TREC10.

We evaluated also our three runs thanks to the patterns given by NIST. The number of right answers is given in the first column of Table 3. Since this evaluation takes into account neither the unsupported answers nor the inexact, these results are quite better. Furthermore, it was interesting for us to look at the results given separately by each strategy, that is to say, without processing the final comparison and selection presented in the preceding paragraph. These are given in column 2, which contains the right answers at first rank, while the third column contains the number of right answers at another rank (between second and fifth rank).

Right answers	Right answers at first rank	Right answers at another rank
Run 1 : 152	Nist1: 128	65
Run 2 : 155	Nist2: 132	66
Run 3 : 165	MG: 136	56
	Web: 122	55

Table 3: Right answers at different rank and for different strategies

Comparing the two first columns, we notice that our choice for an architecture maintaining two different search strategies until the final selection, was a good choice. Indeed, it is obvious that returning systematically the first answer of this set is not without any risk: right answers may often be found between second and fifth rank. Even if the two algorithms used in the final selection can yet be improved, we think they are a promising suggestion for the selection step.

6. Conclusion

TREC11 QA track introduced a new evaluation criterion giving even more importance to the reliability of answers. Indeed, weighting answers is always of great consequence because it determines the answers ranking, but it is particularly important in this case. Table 3, in section 5, shows that a largest number of correct answers are found at the top five ranks, and particularly at first rank (from about 66% to 70%). Actually, we made a real endeavour to introduce a number of weighting criteria at three stages of QALC processing: first, weights assigned to selected document sentences, then potential answers weighting during the extraction process, and finally weights assigned to redundancy of answers retrieved from two corpora, TREC and Web. In such a weighting strategy, the difficulty is to balance the relative weights provided by the different criteria. Weighting criteria that we used are from different kind: lexical, syntactical, semantic and statistical. Latter on, additional criteria based on syntactical dependencies will have to be added.

References

Ferret O., Grau B., Hurault-Plantet M., Illouz G. and Jacquemin C. (2001). *Terminological variants for document selection and question/answer matching*, ACL 2001 Workshop on Open-Domain Question Answering, Toulouse, France.

Fillmore C., Baker C., Sato H. (2002). *Seeing arguments through Transparent Structures.* LREC 2002, Las Palmas de Gran Canaria, Spain, pp. 787-791.

Jacquemin, C. (2001). *Spotting and Discovering Terms through NLP*. Cambridge, MA: MIT Press.

Schmid, H. (1999). Improvements in Part-of-Speech Tagging with an Application To German. In Armstrong, S., Chuch, K. W., Isabelle, P., Tzoukermann, E. & Yarowski, D. (Eds.), *Natural Language Processing Using Very Large Corpora*, Dordrecht: Kluwer Academic Publisher.

Soubbotin M.M., Soubbotin S.M. (2001). *Patterns of Potential Answer Expressions as Clues to the Right Answers.* TREC 2001 Notebook, Gaithersburg, USA, pp.175-182.

TREC 2002 Video Track Experiments at MediaTeam Oulu and VTT

Mika Rautiainen[†], **Jani Penttilä**[‡], **Dmitri Vorobiev**[†], **Kai Noponen**[†], **Pertti Väyrynen**[†], **Matti Hosio**[†], **Esa Matinmikko**[†], **Satu-Marja Mäkelä**[‡], **Johannes Peltola**[‡], **Timo Ojala**[†] **and Tapio Seppänen**[†]

[†] MediaTeam Oulu
P.O.BOX 4500, FIN-90014 University of Oulu, Finland
{firstname.lastname@ee.oulu.fi}

[‡] VTT Technical Research Centre of Finland
P.O. Box 1100, Kaitoväylä 1, FIN-90571 Oulu, Finland
{firstname.lastname@vtt.fi)

Abstract

In TREC 2002 Video Track MediaTeam Oulu and VTT Technical Research Centre of Finland participated jointly in semantic feature extraction, manual search and interactive search tasks. In the semantic feature extraction task, we sent results for semantic categories of cityscape, landscape, people, speech and instrumental sound. Spatio-temporal correlation of oriented gradient occurrences was used with example shots to detect shots containing people, cityscape or landscape. The audio signal features consisted of various statistical measurements and were used to detect shots containing speech or instrumental sound. Our video browsing and retrieval system, VIRE, was used for manual and interactive search tasks. Our system offers two techniques for video retrieval: 1. Multi-modal indexing based on self-organizing feature maps with semantic filtering. 2. An interactive navigating tool that combines two inter-shot properties, temporal coherency and metric similarities, into a view where database shots are presented in a lattice structure. We tested our interactive navigating tool with eight persons to obtain results for 25 pre-defined search topics. In this paper we give an overview of the approaches and a summary of the results.

1 Introduction

In this paper, we first describe our work in semantic feature detection. Then we introduce our video browsing and retrieval system VIRE and employ it in manual and interactive search tasks. In the end of the paper concluding remarks are given.

2 Semantic Feature Detection

2.1 People, Cityscape and Landscape

We evaluated the efficiency of visual features in the detection of people, cityscape and landscape from video shots. Since the video material contained a lot of narration in the audio signal, audio features were omitted from this test.

A well known strategy in city vs. landscape classification of images is to use image's edge gradients [19]. The same approach can be used to discriminate natural structures from non-natural. For example, man-made structures can be distinguished from natural views by computing edge histograms that cumulate in specific orientations when structures are non-natural containing many straight edges. Natural views have more evenly distributed histogram of different orientations.

Our Temporal Gradient Correlogram (TGC) feature computes local correlations of specific edge orientations producing an autocorrelogram, whose elements correspond to probabilities of edge directions occurring at particular spatial distances. The feature is computed over 20 video frames sampled evenly over the duration of video shot. Due to temporal sampling, autocorrelogram is able to capture also temporal changes in spatial edge orientations. From each sample frame, the edge orientations are quantized into four segments depending on their orientation being horizontal, vertical or either of the diagonal directions.

To make the feature vector more discriminative, we used detection of skin-colored local regions to generate four values describing the relative size and structure of consistent skin areas in a video shot frame. First value was *relative amount*, which was simple discrete value between 1 and 5 describing the relative amount of skin-colored local regions. Next values indicated number of *long*, *medium* and *short zero runs* (adjacent non-skin blocks) between skin-colored local regions measuring the uniformity of skin structured areas. To detect skin-colored regions, we marked manually skin areas into 40 key frames selected from the shots in feature development collection and trained a self-organizing map into a skin detector using localized HSV color histogram feature. The histogram was localized into a sector area that covers the typical colors of skin in HSV color space. The sampled local regions of 10x10 pixels were used in localized histogram computation. Degraded quality of test videos was prominent: some of them appeared closely monochromatic and there were large color variances between different videos. Due to this, the prognosis for the success of skin detector was initially set low. However, the feature values were normalized and joined with TGC to examine the discriminative power of less-than-adequately performing skin detector.

To find the shots consisting of a certain semantic concept, we selected sets of example shots from the collection of video data for training semantic feature development, 13 example shots were selected for people, and 10 for both cityscape and landscape. These example TGC and TGC+skin feature vectors were compared against the vectors computed from the shots in the test video collection. The dissimilarity between the features in the test collection and in the training set was computed with L1 norm. The resulting set of most similar shots was pruned from duplicates and rank-ordered to create the final list of shots most probable to contain the semantic feature in question.

2.2 Speech and Instrumental Music

Features for detecting speech and music were developed by researchers in VTT Technical Research Centre of Finland. The classification of audio signal between speech and music is widely studied [1][2][4]. Our approach was based on kNN classification of discrete audio samples with the k value set to 3.

The extraction of confidence for a shot to contain speech or music was derived from the weighted average of speech/music classification results for the discrete 3 second portions of the signal.

The classification between speech and music was computationally very inexpensive, and was determined using only four power-related features. A three-second window of the signal was divided into frames of 50 ms overlapping by 10 ms, and the power inside every frame was calculated. The four used features were the variance of the frame-by-frame power, and the variance of the first and the second order differentials of the power, and finally, low energy ratio [5], which is computed as the percentage of 50ms frames with RMS power less than the threshold-percentage of the mean RMS power. A threshold level of 20% for low energy ratio was found to give best results, and the spread of the four features was increased by log transformations. In the training stage the features were normalized to zero-average and unit standard deviation. The translation and scaling parameters for each feature were stored and used in the classification stage to normalize the test signal.

The audio database used to train the system was assembled by sampling music from a vast assortment of CD's and by using a digital recorder to sample speech from Finnish radio broadcasts. All the samples were then converted to 22050 Hz mono. Both conversational speech and single speaker sections were used. The database also contained both male and female speech, sampled from several speakers. Music from various styles and genres was also included in the training set. All samples were 15 seconds long and the length of the whole database was about 20 minutes for speech and 40 minutes for music.

The classification results of 3 second segments were presented as low-pass filtered time series. Low-pass filtering reduced the effect of separate classification errors and smoothed the transition points between longer segments of speech and music. In addition, mixed signals (containing both speech and music) that would produce a fluctuating series of classification results with a traditional binary decision classifier, are now presented as 'gray' areas that belong to both classes. The new trail of annotation labels shows the degree of certainty of belonging to either class for each three-second audio segment at a time. The numerical results were scaled between 0 and 1, and the weighted mean of the classifications inside each shot was used as the relevance measure for instrumental sound detection. The relevance for speech detection was determined as the inverse of the measure for instrumental sound, so that the sum of these values was always 1.

2.3 Results

The results show that TGC seemed to be most efficient in detecting cityscapes, which may be a result of more structured type of imagery in typical cityscape scenes. Another observation is the degrading effect of poor skin feature detection in the results. Even the detection of people was better using plain TGC, which points out the challenges of using color information in low-saturated and monochromatic videos. The detection of speech and instrumental sounds from audio signal had an averaged precision of 0.641 which was over two times higher than the average precision of 0.246 in people, city and landscape categories. The average precisions for detecting semantic features are shown in Table 1.

Table 1. Average precisions for different semantic features

	TGC	TGC+SKIN	AUDIO
People	0.248	0.168	-
Cityscape	0.299	0.197	-
Landscape	0.193	0.128	-
Speech	-	-	0.645
Inst. Sound	-	-	0.637

3 Manual and Interactive Search Tasks

3.1 Video Browsing and Retrieval System - VIRE

Our video browsing and retrieval system VIRE was used in manual and interactive search experiments. It is based on Java code and can be run on both SUN Solaris and Windows 2000 operating systems. System uses J2SE, QuickTime 6 for Java, WordNet dictionary and MySQL JDBC. The system consists of server and client applications, where server controls the querying through self-organizing feature index maps and provides the client query results. Client software offers two views, one for constructing video queries and another for browsing the database shots. References to physical media data and additional feature information are stored in MySQL-database. Figure 1 depicts the architecture of the VIRE system. The browsing view of the client offers content-oriented parallel navigation through database videos. The browsing procedure would be exhausting without efficient indexing structure. By utilizing self-organizing feature maps we were able to unravel the problems with computational requirements.

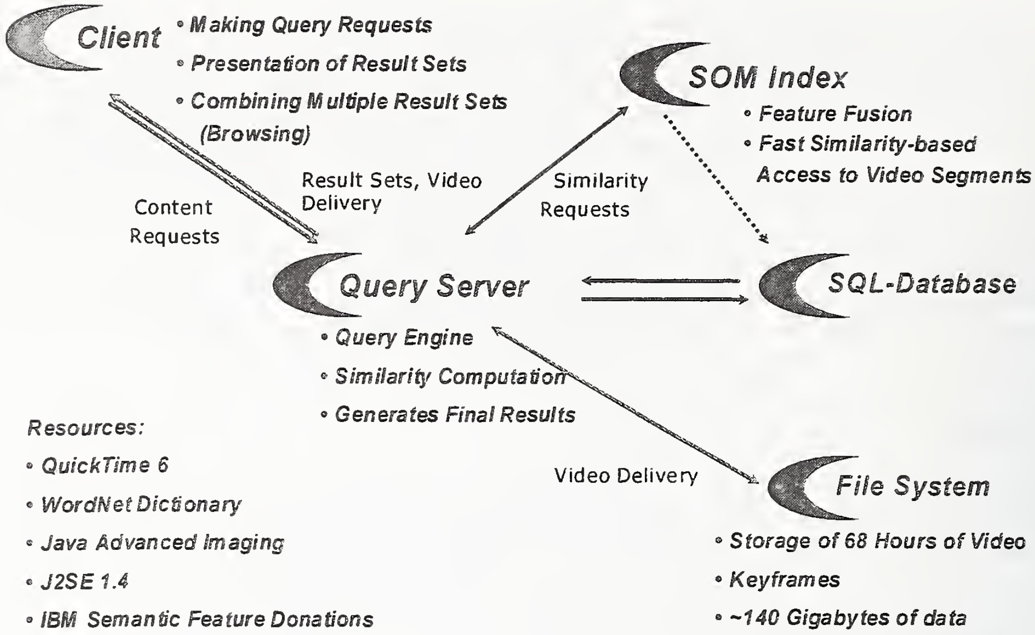


Figure 1. The overall architecture of VIRE, video browsing and retrieval system

3.1.1 Selected Features

Our system uses variety of features, which are used as the components for query. Features are based on different video modalities.

Generic Features

Generic features are measured from the physical properties of video data. Generic feature vectors of query and database shots are compared using distance metrics to measure dissimilarities. Different generic features can be used jointly to measure various physical properties simultaneously.

Color properties of a video shot were described with the Temporal Color Correlogram (TCC). Its efficiency against other color descriptors has been proven in [6][7]. TCC captures the correlation of HSV color pixel values in spatio-temporal neighborhoods. The feature captures temporal dispersion or congregation of color clusters unlike static color features. The parameters we used for the feature are described in [7]. TCC is computed over 20 video frames that are sampled evenly over a video shot.

Motion is a prominent property in video frame sequence. We computed features describing motion activity, based on definitions in MPEG-7 Visual standard [8]. Motion Activity descriptor defines following properties of motion: intensity, direction, spatial and temporal distribution of motion activity. Our system uses the following subset of those features:

- Intensity of motion is a discrete value where high intensity indicates high activity and vice versa. Intensity is defined as the variance of motion vector magnitudes normalized by the frame resolution and quantized in the range between 1 and 5.
- Average intensity measures the average length of all macroblock flow vectors.
- Spatial distribution of activity indicates whether the activity is scattered across many regions or in one large region. This is achieved measuring the short, medium and long runs of zeros that provide

information about the size and number of moving objects in the scene. Each attribute is extracted from the thresholded flow vectors obtained from the video data. All feature values are normalized so that they can be used jointly with other features in self-organizing indexes.

Audio contains a lot of information about the video shot content. And not just the spoken lexicon, but sounds and noises pinpoint details about video's semantic content. The real challenge here is however to choose meaningful parameters which somehow reflect the response to the various properties of sounds in the human aural perception. There has been extensive research on this topic (see [9] and references therein) primarily in connection with speech recognition systems [10]. We have selected following features to construct a generic audio feature descriptor [12][13]:

- Zero-crossing rate (ZCR), one of the most commonly used audio features, gives information about the spectral content of the signal. Taking into account relative spectral deficiency of the sound tracks of the movies in the VideoTREC database (which is obviously the result of their relatively old age) our measurements showed us that the bandwidth of a typical sound track was about 8 kHz. We feel that it is safe to assume that ZCR tracks the fundamental frequency f0 with quite high precision [9] due to the absence of high-frequency components. Some researchers believe [13] that ZCR is one of the most indicative and robust measures to discern unvoiced speech.
- RMS energy measures mean signal energy, which is what the human ear interprets as a notion of the acoustic volume, but it does not carry any information about the presence of transient sounds.
- Maximal and minimal energy are two parameters that correspond to the idea of "still" and "loud" sound, respectively. If the minimal energy is large and close to the value of the maximal energy, one can attribute the property of acoustic loudness to the entire sound clip. On the other hand, low maximal energy is a reliable indicator of a silent sound clip and hence it is useful for quick and easy silence detection.
- Mean and standard deviation sample energy gives a measure for scattering of the sample energies about their mean value. [11]
- Percentage of samples whose energy is below 50% of the mean energy of the sound clip has been used for the purpose of speech/music discrimination [14]. The fact is that the speech signal contains more "quiet" frames so this value will be higher for speech than for music.
- Percentage of samples whose energy is below 10% of the mean energy of the sound clip might carry some information about transient sounds and the purpose of using this feature was to improve scene detection. If the mean energy is significantly greater than that of the majority of sound samples, it might indicate the presence of short-term shooting-like acoustic events.
- Harmonicity ratio describes the proportion of harmonic components in the spectrum. The algorithm output is 1 for purely periodic signal and 0 for the white noise. A useful feature derived from harmonicity ratio was the length of the comb filter, which is an estimate of the delay that maximizes the autocorrelation function.
- Upper limit of harmonicity loosely defines the frequency beyond which the spectrum has no harmonic components. These features were calculated in 2048 sample windows that overlapped by 1024 samples. All audio files had sampling frequency of 22050.
- Spectral centroid is the center of gravity of the power spectrum. For audio signals that have clearly much energy in lower or higher parts of the spectrum this feature is useful.
- Spectral spread is the RMS deviation of the spectrum centroid, and thereby it describes if the spectrum is widely spread out or concentrated around its centroid. The used values were median values in one second window. These features were calculated in non-overlapping windows of 1024 samples.

To compute the dissimilarities of generic feature vectors, we have used L1 norm. Each generic feature vector is normalized prior inserting into self-organized index.

Semantic Features

Semantic features are single lexical concepts exist in a shot with certain confidence. Our system utilizes these concepts as binary filters to create subsets from results obtained with fuzzy generic feature dissimilarities. Different combinations of concepts can be used to create filter sets. In this case, the resulting subset will be the intersection of existing concepts in the initial result set.

We used following IBM donated semantic features: face, people, indoors, outdoors, instrumental sound, speech, landscape and text. These features had a confidence value that was thresholded into binary filter rules. The threshold value was decided upon criteria where approximately one third of the shots were assigned with the concept. Following threshold values were used: 0.62 for face, 0.75 for indoors, 0.35 for instrumental sound, 0.7 for landscape, outdoors, and people, 0.4 for speech, and 0.3 for text.

Text Features

Text features are derived from the automatic speech recognition (ASR) data that was made available for all participants by CLIPS-IMAG. The textual information was not used as a feature vector. Instead, it was used like semantic features as a filter for the results acquired from SOM-index structure. It was used to eliminate shots that did not contain indicated textual terms. To accomplish this, the ASR transcripts were indexed into a database treating the words as single-word terms. A stop word list was used to exclude grammatical and otherwise undiscriminating words that would have led to poor resolution. Qualified words were then lemmatized using the morphological processing features of the WordNet [15].

Because of the effects of lexical semantic phenomena such as homonymy or polysemy on the relevancy of the documents retrieved, techniques of word sense disambiguation (WSD) have been found useful in information retrieval to mitigate the effects of expressive power of natural language. Potentially relevant documents containing close synonyms of the query word such as 'dog' and 'canine' and hyponyms such as 'Malamute' will be missed unless queries are expanded by synonyms and hyponyms of the terms used in original user requests. [16]

To evaluate queries, we used relevance metrics to measure which shots were suitable given a set of topic related query words that were synonym expanded. The ranking of the shots was computed using Term Frequency Inverse Document Frequency (TDIDF) [17] based classification method that pinpoints relevant single-word terms occurring in the ASR transcripts. First, the given query words were automatically reduced to their base form. Second, the lemmatized query words were expanded with their synonyms using the WordNet [15]. Third, the relevance metric was computed for every shot that contained at least one of the words in the expanded query set. Finally, the neighbors of suitable shots were also included into the results within a time frame of 4 seconds. This was done for the sake of the temporal locality of the topic that spans over short shots that can hardly contain any ASR information.

Our approach encounter challenges as the video test material consisted of degraded audio quality, which seemed often lead to false detections of words. To use such data in a rather restricting filter yields inflexibility in the cases of erroneous interpretation of the spoken words.

3.1.2 Multi-modal Indexing Based on Self-organizing Maps

To avoid exhaustive searching on the server side, we used self-organizing index maps that are capable of finding metrically closest matches with any feature combination within tight timing requirements set by interactive video browsing that requires much parallel query processing. Our index structure consists of seven self-organizing maps (SOMs). Three of the SOMs are based on the primary generic features: color, audio and motion. Next three SOMs compose of joint features from the primary generic features: color&audio, color&motion and audio&motion. Last SOM uses all primary features: color&audio&motion. Each of the SOMs is generated with the SOM parameters set in the SOM Toolbox [18].

All the examples in a single query are processed individually and access to specific index maps is directed according the selected set of features. Each individual example launches best matching nodes –search returning a set of closest shots from the appropriate SOM.

Next these intermediate results are filtered using semantic feature or text description filters. Multiple filters can be selected, for example 'Outdoors', 'People' and text 'red Chevrolet'. The result sets are evaluated based on the existence of these filter terms and only the shots fulfilling the criteria are selected. Then, these sets of results are combined with fuzzy Boolean OR operator to form the final ranked result set. Different weights can be set to examples to change the order in the final, combined results. The selection of the amount of best matching nodes is guided by the source of the query. In browsing, the speed is emphasized over retrieval precision, so the number of best matching nodes is small. We have used 3, 10 and 70 best matching nodes in fast browsing, more precise browsing and manual querying, respectively. Figure 2 illustrates the indexing structure.

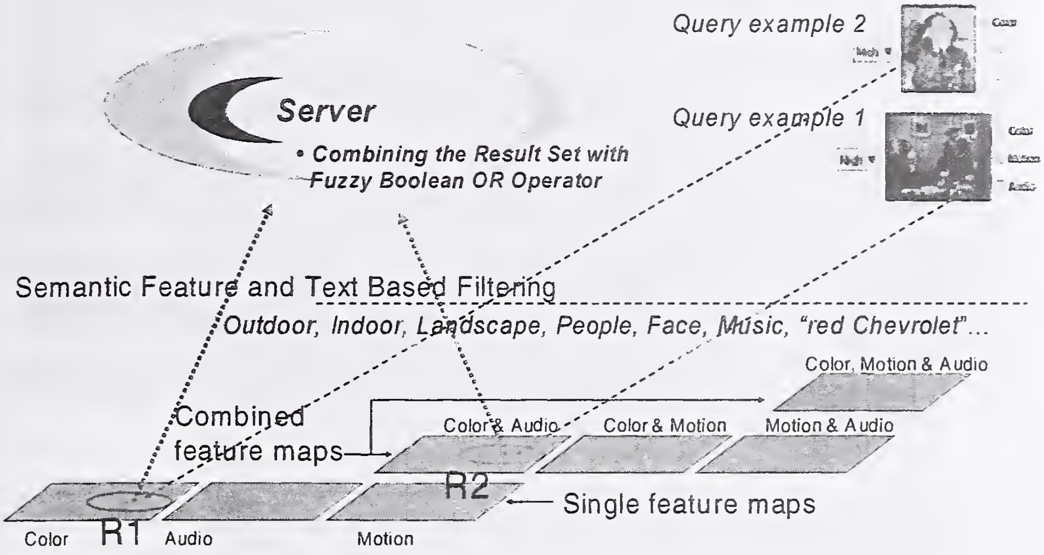

Query example 2
Query example 1
Server
• Combining the Result Set with Fuzzy Boolean OR Operator
Semantic Feature and Text Based Filtering
Outdoor, Indoor, Landscape, People, Face, Music, "red Chevrolet"...
Color, Motion & Audio
Combined feature maps
Color & Audio
Color & Motion
Motion & Audio
R2
Single feature maps
Color
R1
Audio
Motion

Figure 2. Self-organizing index structure organizes database shots into various maps.

3.1.3 Manual Query Interface

Figure 3 shows the query interface devised for manual search. The view offers selections of various search features and filters in the left side and resulting shots ranked by their similarity with the selected query attributes are displayed in the right side view. Query topics can be changed from the Topic menu. This will change the example shots or images provided by the topic description to the lower left panel.

User can select any combination of three generic features (color, motion or audio) for any example shot, but for example image only color feature is supported. User can enable any set of semantic filters in addition to the selected generic features. Additionally, a lexical word filter can be constructed by selecting a set of words in the upper left panel of the interface. At least one example shot with one generic feature enabled is the minimum requirement for submitting a query. From the result set, user can select any interesting shot as a start point for navigation in the browsing interface.

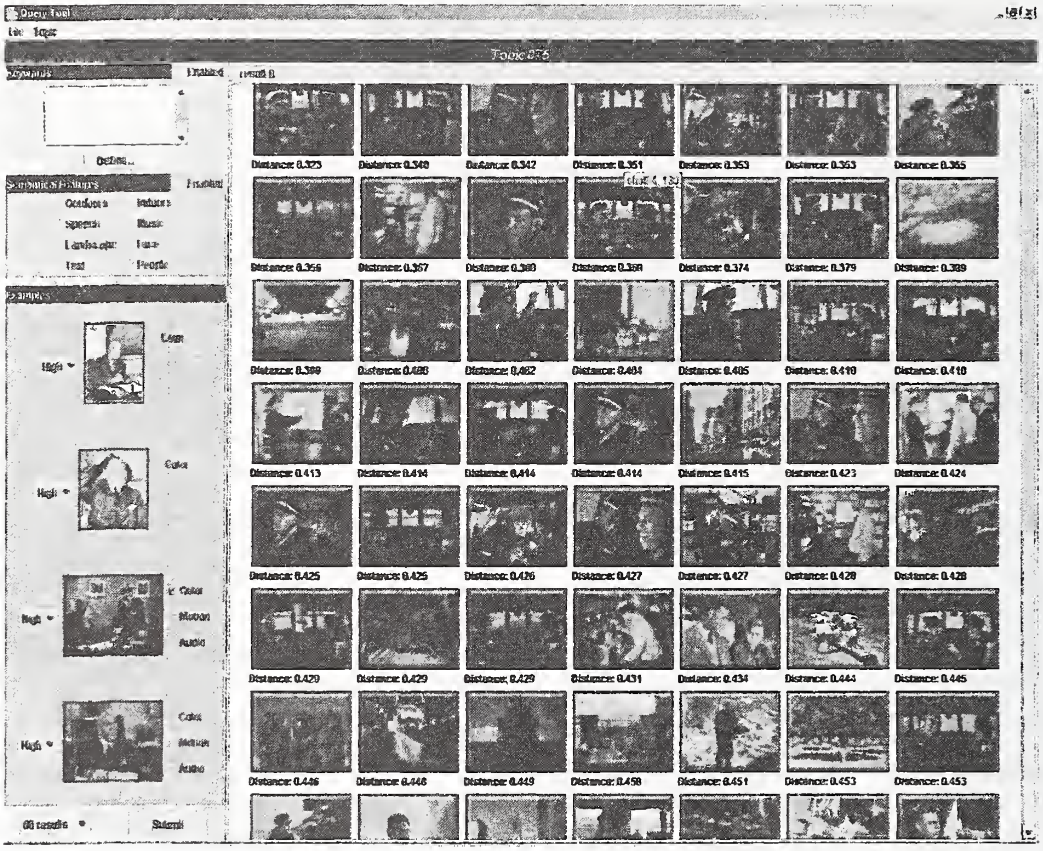


Figure 3. Manual query interface

3.1.4 Interactive Browsing Interface

The novelty of our approach in the interactive search task relies on two aspects: interface design and content-based feature processing. The motivation is to reduce the effect caused by ambiguous results that are usually obtained from a traditional content-based example search. Currently, two dominant approaches are used to realize video searching and browsing. Systems either select content-based presentation of video items or rely on more traditional time-line based organization into temporally adjacent items. The disadvantages of the approaches are in their incapability to associate computed features with the user's information need (ambiguity of content-based approaches) and to provide a holistic view over the linear temporal presentation (inefficiency of the time-line based browsing).

Our approach combines both inter-video similarities and local temporal relations of video shots in a single interface. In interactive search, users want the computer to act as a 'humble servant', providing enough cues and dimensions for users to navigate through the vast search space towards the relevant objects. In VIRE, users can perform *content-oriented browsing* that combines timeline presentation of videos with content-based retrieval. Content-oriented browsing implies that the video content is not utilized alone, but in conjunction with temporal video structure.

Figure 3 illustrates the browsing interface. The panel showing the first row of key frame images displays sequential shots from a single video in a chronological time-line. At any time, user can scroll through the entire video shot sequence to get an overview of the video content. The leftmost key frame in the top row shows always the first shot and name of the video. The first shot may contain initial setup for the entire video, so by viewing it, user can get instant idea about the semantic setting for the rest of the video shots.

The lower right panel gives user another content-oriented view, but this time from the entire database. The columns below the topmost shots show the most similar matches organized in top-down rank-order. The columns generate a similarity lattice that provides linkage to other database videos. The similarity is measured based on the features selected in the lower left panel. User can select a single feature or any combination depending on what properties they want to browse with.

Additionally, user can decide whether he want to include shots from the query video to be shown on a similarity lattice. When user locates interesting shots from the lattice, he can open the video in the topmost row so that the interesting shot is located in the middle column. After updating the shots in the topmost row, system re-computes the similarity lattice. At any time, user can update the current lattice using other feature combinations. The features that can be used in browsing are color, motion, audio and texture.

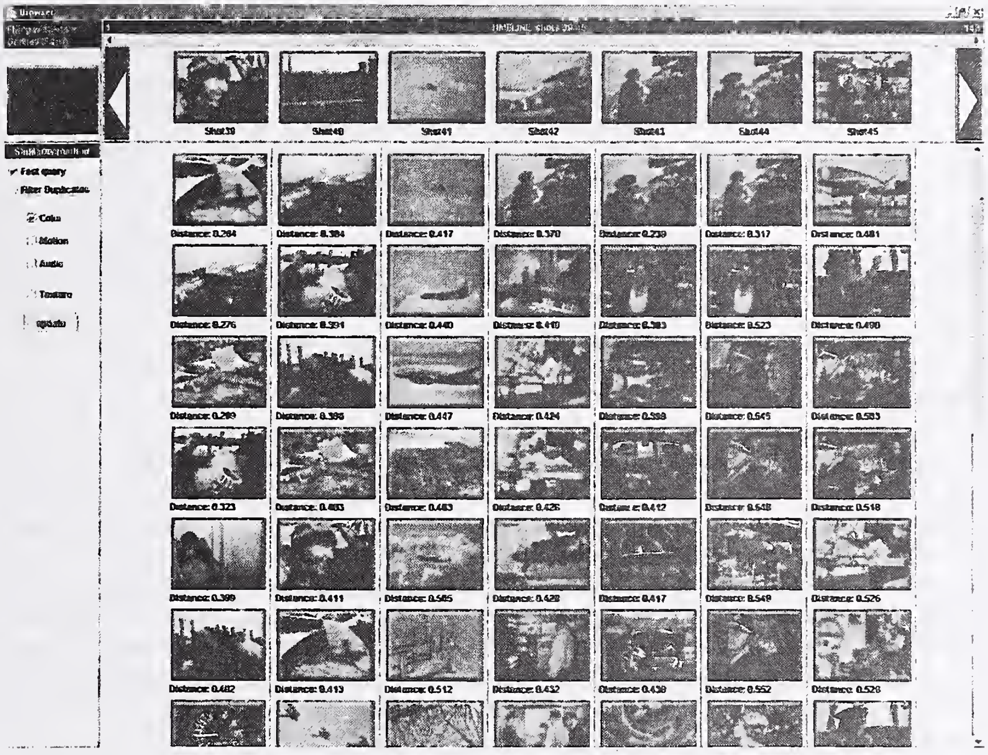


Figure 3. Content-oriented browsing interface. Lower right panel shows the similarity lattice.

The requirements to update the similarity lattice are heavy, since the browsing speed should be close to real-time. To update a single lattice, system must perform parallel query processing in several individual example-based queries. Multi-threaded index queries to self-organizing maps provide efficient access mechanism to proximity search for every feature combination.

3.2 Experimental Setup

We tested our system in both manual and interactive search. NIST provided 25 search topics that contained varied from very specific ('Find shots with Eddie Rickenbacker') to more generic concepts ('Find overhead views of cities'). Topic included one or more example clips of video or images to aid the search process. From the image examples, only a color correlogram could be used as a generic search feature whereas the video examples offer color, motion and audio or any of their combinations. All topics were processed both manually and interactively in search test video collection that consisted 40 hours of video material. NIST also provided segmentation for all videos with more than 24000 video shots, from which over 14000 belonged to the search test collection.

In manual query the user initiated the query by selecting appropriate generic feature cues from the topic examples, sets weight to indicate relevant and non-relevant examples and decides appropriate filters. Three different search configurations were tried for each search topic: search using generic features only, search using generic features with semantic feature filtering, and search using generic features with text feature filtering. Self-organized index was used with 70 best matching nodes.

A group of eight new users carried out the interactive search. Test users, most of them males, were information engineering undergraduate students, having good skills in using computers, but less experience in searching video databases (obviously at least somewhat experienced in www-search). Every user reported to be somewhat familiar with the search topics. 25 topics were divided into four sets that were randomly given to the test users so that two users carried out the same set of topics. All users were given half an hour introduction to the system, with emphasis on the search and browsing interface functions demonstrated with couple of example search. Users were told to use approximately ten minutes for each search, during which they navigated in the shot database and selected shots that seemed to fit to the topic description. Users were also told to fill a questionnaire about their experiences. The machines that the system was running on were 400-800 MHz PCs with Windows 2000 operating system. After the tests were finished, the two result sets for one topic were joined by removing duplicate matches and averaging their rank values.

3.3 Results

Average precisions for the four different search configurations are shown in Table 2. Manual configurations used either only Generic Features (color, motion or audio), Generic Features together with semantic feature filtering (outdoors, landscape) or Generic Features together with text filter ("red Chevrolet"). Number of hits at depth 10 describes the total amount of found matches when considering the 10 best ranked matches for a topic. Overall average shows the mean value of the average precisions in 25 topics. As can be seen, the performance of interactive search overcomes greatly the manual search results. This indicates clearly the importance of the human factor in search. The average time in making an interactive browsing was 10.08 minutes. During this time, the average of 12.7 matches were found from the database. This means that average of 1.26 matches were found during each minute of searching. Another interesting observation is the counter-effectiveness of semantic and text filters to the results.

Table 2. Results for different configurations over 25 search topics

Type of search configuration	Nr. of hits at depth 10	Overall Average
Interactive	151	0.26
Manual (Generic Features)	38	0.03
Manual (Gen. Features + Sem. Filters)	22	0.02
Manual (Gen. Features + Text Filter)	12	0.01

The six most successful topics are listed in Table 3, best resulting topics being topmost. Finding George Washington, Price Tower and James H. Chandler overall seemed to be the most successful topics in manual and interactive searching across the participating system runs.

Table 3. Six most successful topics (topic number in parenthesis)

Interactive Search	Manual Search (Generic Features)
James H. Chandler (76)	James H. Chandler (76)
George Washington (77)	Microscopic living cells (97)
Parrots (91)	Eddie Rickenbacker (75)
Price Tower in Oklahoma (84)	Musicians (80)
Microscopic living cells (97)	Snow covered mountains (90)
Nuclear explosion (95)	Overhead view of cities (86)

According to the answers in questionnaire, the VIRE system was easy to learn, but somewhat harder to use. This was due to the ambiguous results that fuzzy-based similarity measurements return in many instances. The browsing interface was appreciated, although the near-real time responses in updating the browsing view would have been preferred to be completely real-time.

4 Conclusions

We approached content-based video retrieval from many different perspectives in TREC 2002 Video Track. Our experiments showed that the extraction of semantic concepts is a challenging task. However, the results in detecting instrumental sound and speech were promising. Visual semantic features have still a lot to improve: the low visual quality of the videos definitely affected the results, though.

The most successful component in our work was the content-oriented browsing that gave the computer merely the role of an assistant in the cognitive process of semantic searching. Our system was subordinated to provide the user multiple parallel paths from which he could choose the direction for his navigation independently. We combined the temporal connectivity of temporally adjacent shots and fuzzy shot similarities into one view to provide the user comprehensive information about the inter-relations between the video shots.

The manual search methods must still overcome big challenges to reach a satisfactory level of performance in the high-level semantic search problems. However, there are search problems that are more suitable to automatic search than others, for example locating cityscape views or retrieving shots containing speech. It seems that while efficient features can be computed from different modalities, they are not alone appropriate for automated semantic retrieval. Does the missing link lie within a single feature quality, or rather in a way to combine multiple modalities? This still remains an intriguing research problem.

5 References

[1] Carey M, Parris E & Lloyd-Thomas H (1999) A comparison of features for speech, music discrimination. Proc. ICASSP.

[2] Hoyt J & Wechsler H (1994) Detection of human speech in structured noise. Proc. ICASSP.

[3] Kedem B (1986) Spectral analysis and discrimination by zero-crossings. Proc. IEEE. Vol. 74, No. 11.

[4] Saunders J (1996) Real-time discrimination of broadcast speech/music. Proc. ICASSP.

[5] Scheirer E & Slaney M (1997) Construction and evaluation of a robust multifeature speech/music discriminator. Proc. ICASSP.

[6] Ojala T, Rautiainen M, Matinmikko E & Aittola M (2001) Semantic image retrieval with HSV correlograms. Proc. 12th Scandinavian Conference on Image Analysis, Bergen, Norway, pp.621-627.

[7] Rautiainen M & Doermann D (2002) Temporal color correlograms for video retrieval. Proc. 16th International Conference on Pattern Recognition, Quebec, Canada.

[8] MPEG-7 standard: ISO/IEC FDIS 15938-3 Information Technology - Multimedia Content Description Interface - Part 3: Visual.

[9] Gerhard D (2000) Audio signal classification, School of Computer Science, Simon Fraser University, Burnaby, Canada.

[10] Rabiner L & Juang B-H (1993) Fundamentals of speech recognition, Prentice Hall.

[11] Gray R & Davisson L (1985) Random processes: a mathematical approach for engineers, Prentice-Hall Information and System Sciences Series, Prentice Hall.

[12] MPEG-7 standard: ISO/IEC FDIS 15938-4: Information technology -- Multimedia content description interface -- Part 4: Audio

[13] Wang Y, Liu Z & Huang JC (2000) Multimedia content analysis, IEEE Signal Processing Magazine, November 2000, pp.12—36.

[14] Scheier E & Slaney M (1997) Construction and evaluation of a robust multifeature speech/music discriminator, Proc. ICASSP-97, April 21-24, Munich, Germany.

[15] Fellbaum C (1998) WordNet: An electronic lexical database. The MIT Press.

[16] Jurafsky D & Martin JH (2000) Speech and language processing: an introduction to natural language processing. Computational Linguistics, and Speech Recognition. Prentice-Hall.

[17] Salton G & Yang C (1973) On the specification of term values in automatic indexing. Journal of Documentation, Vol. 29, 351–372.

[18] SOM Toolbox (2.11.2002) http://www.cis.hut.fi/projects/somtoolbox/documentation/somalg.shtml

[19] Vailaya A, Jain A & Zhang HJ (1998) On image classification: city vs. landscape. IEEE Workshop on Content-Based Access of Image and Video Libraries, pp. 3-8.

TREC Feature Extraction by Active Learning

J. Vendrig[1] J. den Hartog[2] D. van Leeuwen[3]
I. Patras[1] S. Raaijmakers[2] J. van Rest[2] C. Snoek[1]
M. Worring[1]

[1]Mediamill/ISIS, Informatics Institute, University of Amsterdam, Kruislaan 403.
1098 SJ Amsterdam, The Netherlands
[2]Mediamill/TNO-TPD, Netherlands Organisation for Applied Scientific Research.
Stieltjesweg 1, 2628 CK Delft, The Netherlands
[3]TNO-Human Factors. Netherlands Organisation for Applied Scientific Research,
Kampweg 5, 3769 ZG Soesterberg, The Netherlands

1 Introduction

Current multimedia retrieval research can be divided roughly into two camps. One camp is looking for the panacea which solves all problems in one system. The other camp focuses on very specific problems in restricted domains. In our opinion. the answer lies in the middle. A system should not desire to solve all problems, but should take advantage of a user's knowledge about his or her specific problem, so that the system can focus on it. On the other hand, available video analysis techniques should be extended to other domains which possibly were not envisioned upon their design. The challenge is transparent application of video analysis techniques to the appropriate user domains.

A user, especially an expert, has best knowledge about the characteristics of a particular domain. In this paper the user's input is given at index-time. rather than at query-time as done in our TREC 2001 contribution [2]. In [2]. we associated user queries with video content descriptors via general Wordnet concepts. For example. query term *woman* maps to Wordnet hypernyms *"person. individual, human"* which we associated with the "face presence" descriptor. In this TREC 2002 contribution, we focus on building models for the association of content descriptors with generic concepts, such as the Wordnet hypernyms. Specifically, we focus on the ten generic concepts given by the TREC feature extraction task[1]. User and machine interact in order to map the semantic feature concept to content descriptors for a training set, so that shots can be classified for use in retrieval applications.

In this paper, we assume that every feature model is specific to not only the domain, but even to a collection. in order to exploit domain characteristics

[1] In the remainder of the paper we follow the TREC terminology, referring to the semantic concepts as features.

and user domain knowledge. The use of a small number of broadly applicable features for video content classification is described by IBM [11] in last year's TREC, showing relatively good results. Their probabilistic system mixes the use of models for general features (e.g. *outdoors* and *face*) with models for domain-specific features (e.g. *rocket* and *fire*). In our approach, every model is collection-specific. That is, even a general feature is considered to be specific. For example, although the *outdoors* semantic concept can be found in many video collections, its visual representation in a video may be quite different. Footage of the Discovery Channel shows a different kind of outdoors sceneries than television sitcoms. The old instructional videos for school kids in the TREC 2002 collection are quite different from the videos shown at school to the MTV- and Nintendo-genérations. In addition, a feature can be defined differently amongst domains, resulting in the need for different models as well. For example, the definition of the *face* feature for a Cartoon Network collection is different from the one for C-SPAN.

Focusing on one particular collection enables for use, or some might say abuse, of simple content descriptors that are correlated to a semantic concept. This may be caused by the style of one or more people in the film crew, such as director, editor and camera man. An example of specific collection characteristics is found in the TREC 2001 video collection. The camera movement descriptor could be used to classify shots as mountains, as they roughly correspond with camera pans. Although such a classification method cannot be generalized to other collections, it allows for uncomplicated retrieval of videos in a specific collection.

We present a system which interactively learns user-defined semantic concepts for a specific collection from a domain expert. For each concept, the domain expert builds a model by feeding visual evidence to the system in the form of examples, without knowledge about the underlying classifier and descriptors. We employ a large set of multimedia descriptors for use in a Maximum Entropy classifier. The space for example selection is determined by the output of the incrementally improved model. The system is evaluated against the TREC 2002 feature extraction collection. The user information consists of the ten semantic concepts defined for the feature extraction task.

As our system is based on visual evidence, we focus on visual content of videos. That is, we focus on the features *outdoors*, *indoors*, *facc*, *people*, *cityscape*, *landscape*, *text overlay* and *monologue*. The classification of the audio features (*speech* and *instrumental sound*) is provided independently and is described briefly in section 2.2.

The paper is organized as follows. In section 2 we describe how video content is represented for example selection and classification. In section 3 the use of the Maximum Entropy classifier for multimedia content is described. In section 4 the interactive selection of examples using active learning is explained. In section 5 we describe the experimental setup for TREC evaluation. Results are discussed in section 6. Finally, we present conclusions and future research in section 7.

2 Content representation

2.1 Data representation

The elementary unit in the context of TREC is a shot. However, the shot itself, i.e. the sequence of frames, is not always a good representation for the visual content. There are two reasons for employing an alternative representation. Firstly, a shot is not necessarily visually and semantically coherent. In [14] such fractions of a shot are called a shot-let, while in [12] this division is referred to as named events, which are short segments with a meaning that does not change in time. Division of the shot into smaller coherent fractions allows for better representation of the shot's content. In the context of TREC the further division is especially important, as the reference shot segmentation suffers from undersegmentation, combining consecutive shots into one shot. In addition, division into lower level units prevents loss of information due to aggregation. This is especially important in the context of TREC, as the feature definitions state that a shot is assigned to a class when at least an observable part of the shot belongs to that class. That is, a shot could be assigned to disjuncts concepts. e.g. both outdoors and indoor.

The second reason is computational feasibility. Using expensive descriptors derived from image processing on each frame in a shot requires a large amount of computing power. Meanwhile, the descriptor values can be expected to be highly similar in consecutive frames, as the content of frames change just gradually within a shot. Therefore it is expected that choosing a representation which requires less computing power does not result in loss of information as a consequence.

We choose to use content-dependent key frames as representation of a shot for image processing based descriptors. Motion descriptors are calculated on shot level for practical reasons, i.e. compatibility with existing systems. Content-dependent selection of key frames is based on the change in visual content during a shot compared to the change of content in the entire video [7]. Shots containing a relatively high amount of changes are assigned more key frames. Within a shot, key frames are chosen such that the total amount of change in the surrounding segment is approximately equal for all key frames in the shot.

2.2 Descriptors

Descriptors describe the content of a shot, or the content of a representation of a shot, in order to enable comparison of the content of two shots. For automatic use by the classifier, we employed a descriptor pool containing over 60 descriptors. The descriptor pool is not geared towards a specific data collection, as the system is designed to be independent of the data collection. The descriptors include atomic descriptors such as color values and color distributions, edge characteristics, and motion descriptors; and complex descriptors such as face presence [10] and camera movement [1]. Due to the large amount of descriptors, it cannot be assumed that they are independent. For example, one descriptor (e.g. dominant color) may coexist with a specialization of itself (e.g. dominant color in the top half).

In the following sections, we describe in more detail the descriptors that relate specifically to the temporal component of video shots.

2.2.1 Motion

Motion descriptors are extracted by a two-level analysis. At the block of frames level we analyze the motion by estimating a parametric motion model with a robust regression scheme [1]. In this way we obtain two types of low-level features. On the one hand we obtain the camera operation (pan, tilt, zoom-in, zoom-out or unknown) and the factors that are related to it (focus of expansion, pan-factor, etc). On the other hand, we obtain descriptors such as the average motion, the percentage of outliers from the dominant motion model, and the average position and motion of the outliers.

At the shot level, the descriptors of the block of frames level are combined for the estimation of descriptors such as the average pan factor in the shot, the motion activity due to camera operations and the percentage of frames in which the camera zooms in. The shot level descriptors are used for classification of the shot.

2.2.2 Audio

The classifications for the two audio features (speech and music) are provided independently by TNO-Human Factors. Classification is done without prior knowledge about the data set.

The speech/music discrimination is based on amplitude variation in the audio's signal envelope shape. Generally, speech has higher amplitude variations than music in the spectral regions around 475 and 2700 Hz. The input signal is partitioned into 1 second segments. For each segment, the amplitude fluctuations in these bands are determined and they are low pass filtered at 8 Hz. When the amplitude variation in either of the two spectral bands is above a certain threshold, the segment is identified as *speech*, otherwise as *music*. A third category *silence* is used if the total acoustic energy is low. The thresholds are based on values found for Dutch radio and television broadcast material, as well as eight music CD tracks with various music styles (classical instrumental, vocal, pop and jazz).

3 Classifier

The concept classifier used to model features has to deal with three issues concerning the data set and the descriptors. Firstly, the classifier has to cope with descriptors that are undefined for a shot. For example, when there is no pan camera movement in a shot, the pan factor descriptor is undefined. Secondly, in contrast with [11], it is our opinion that the classifier cannot assume descriptors are independent. Independence is not expected in multimedia objects, such as video shots. because they comprise several correlated information sources. In addition, the use of a large descriptor pool leads to interdependencies of descriptors. Thirdly, the classifier has to cope with an imbalanced data set. Just a relatively small number of positive examples is available in the training set. Training should focus on the positive examples. since the given features are one class problems [15].

For classification we choose the Maximum Entropy classifier, as it deals with the above issues. Firstly, it makes use of sparse vector format, thereby dealing with missing descriptor values. It is not necessary to provide dummy values for undefined descriptors. Secondly, the classifier assumes no independence between descriptors, in contrast to classifiers as Naive Bayes [8]. Thirdly, we found in experiments that a Maximum Entropy classifier is less sensitive to the majority effect than a Support Vector classifier [8]. Hence it is well suited for a data collection containing few positive examples.

Maximum Entropy classification has been applied successfully in a variety of domains, including the area of statistical natural language processing where it achieved state-of-the-art performance [4]. The Maximum Entropy framework was originally proposed by Jaynes [9] as a means to make inference based on partial information. Jaynes claimed that "the only unbiased assignment one could make. should use the probability distribution which has maximum entropy subject to whatever is known", i.e. the probability distribution keeping the uncertainty maximal.

The Maximum Entropy approach allows for the use of a large amount of descriptors without the need to specify their relevance for training a specific semantic concept. The relative importance of each descriptor is computed automatically by the *Generalized Iterative Scaling* (GIS) algorithm [6]. This makes Maximum Entropy classification generally applicable.

A general problem for classifiers is diversity in descriptor types. The Maximum Entropy classifier suffers from this problem as well as it makes use of binary trigger descriptors. i.e. either the descriptor is true or it is false/undefined. Our original multimedia descriptors are both categorical and numerical. An example of the former descriptor type is "type of camera movement", which takes values such as "zoom" and "pan". Categorical descriptors can be used directly as a binary trigger. The mapping of numerical descriptor values to binary triggers. however. is non-trivial.

For further discretization of numerical descriptor values we employ a binning function which maps each value to a categorical representation. The binning process itself does not need to lead to loss of information. For many descriptors, numerical values have a higher precision than is needed or used. For example, there is no significant difference between value 0.10 and 0.101, i.e. more precision does not necessarily lead to better description of the content. It is often sufficient to express a descriptor's value categorically. An example is the use of "mostly orange" or "very orange" to describe a color as done in [13]. The choice for the number of categorical values (bins) has to be established experimentally.

The binary trigger descriptor resulting from binning is used by the Maximum Entropy classifier. A disadvantage of binning which is specific to the Maximum

Entropy approach is the loss of order. That is, for the classifier bins are not related in any way. Bin 1 is not closer to bin 3 than to bin 9. Even when binning does not lead to loss of information for an individual descriptor value, it does lead to information loss for the similarity between shots.

The binning function has to take into account that not all descriptor values are normalized. Therefore we choose the use of equal frequency binning, which is performed after descriptor computation for the entire collection. It divides the descriptor values into a fixed number of bins such that each bin contains approximately the same number of values. Equal frequency binning is not sensitive to outlying values and skewed distribution of values over the range. The remaining problem is to choose the number of bins for discretization.

The implementation used for the Maximum Entropy classification is the publicly available OpenNLP Maximum Entropy Package [3].

4 Interactive teaching

As all learning classifiers, Maximum Entropy suffers from the need to select training examples from the collection. It requires a great deal of human effort to label the examples. Minimizing the human effort without compromising the quality of the examples is an important issue.

The traditional approach of random sampling is often chosen to acquire examples from a collection. However, for the problem at hand random sampling does not seem optimal. Cohn [5] addresses this issue in general in the context of neural networks, stating that in many formal problems it is more efficient to focus on a region of uncertainty rather than the entire collection. For the specific problem of TREC 2002 feature extraction we found this to be the case.

The reason to use more intelligent example selection than random sampling lies in the nature of the TREC features. Although the features can be perceived as binary (e.g. a shot contains *overlay text* or not), really the feature can be defined by positive examples only as the scope of negative examples is too broad. Therefore, focus on finding positive examples is needed.

In addition to the relative importance of positive examples, the positive examples are relatively rare in the collection. For example, we estimate 5% of the shots in the training collection contain *overlay text*. Even when a classifier does not need many positive examples, a sufficiently large set in absolute numbers must be given. Hence, it is more important to find positive examples describing the feature than to find examples representative for the collection.

To give precedence to labeling of positive examples we apply active learning, which is defined by [5] as "any form of learning in which the learning program has some control over the inputs it trains on". That is, the classifier controls which examples are presented to the teacher for judgement.

Because of our focus on positive examples, for employment of active learning in our system we use a variant on the region of uncertainty described in [5]. Instead of presenting the teacher shots of which classification is most uncertain, we present shots for which labeling is most important, i.e. the shots most likely to be positive examples. This way, the teacher does not just provide examples, but indirectly he or she gives feedback on the classifier.

Theoretically, the active learning approach described could lead to undersampling of negative examples. However, in practice this would occur only when the model is very good from the start, or when positive examples are abundant in the collection. Both cases are unlikely in the context of the TREC feature extraction task.

Ideally, all presented examples would be classified by the teacher as either positive or negative. In practice, we have to introduce an "unclassified" category for two reasons. The first reason is that some shots do not contain sufficient information to be classified unambiguously. That is, the feature value cannot be determined without speculation or knowledge about the context. The second reason relates to definitions. The definitions given for the TREC experiment do not always match with the intuitive classification for a feature. This problem is not specific to TREC, but would occur in any large data collection for a broad domain, especially when there is more than one teacher. In cases where intuition and strict definition clash, the teacher uses the "unclassified" label. In terms of the classifier it means the model is not trained on "unclassified" examples, and we have no opinion on the outcome of the classifier for such examples when applied to the test collection.

We use the i-Notation system described in [16], extended with access to the OpenNLP Maximum Entropy Package [3] for selecting possibly positive examples. It is depicted in figure 1.

5 Experimental setup

In this section we describe briefly the most important parameters influencing the experiment.

Key frames are selected as described in section 2.1, with a minimum of 2 key frames per shot and an average (per video) of 1 key frame every 2 seconds.

The amount of bins is fixed to 4 for all numeric descriptors in the pool. Experiments with other bin amounts in the range of $3 \ldots 10$ showed no significant impact on results for the training set.

As the Maximum Entropy classifier's decision is binary for each feature, the results consists of shots for which a positive decision with a likelihood ≥ 0.5 is found.

The confidence of the existence of an audio feature (*speech* or *music*) within a video shot is computed as the number of segments classified as the feature normalized over the total shot length. The confidences of all shots are ranked and cut off at the given maximum of 1000 shots.

The two runs are different for the eight visual features only. The two aural features are constant. In the second run, we add the confidence measures for the aural features as descriptors used to classify the eight visual features.

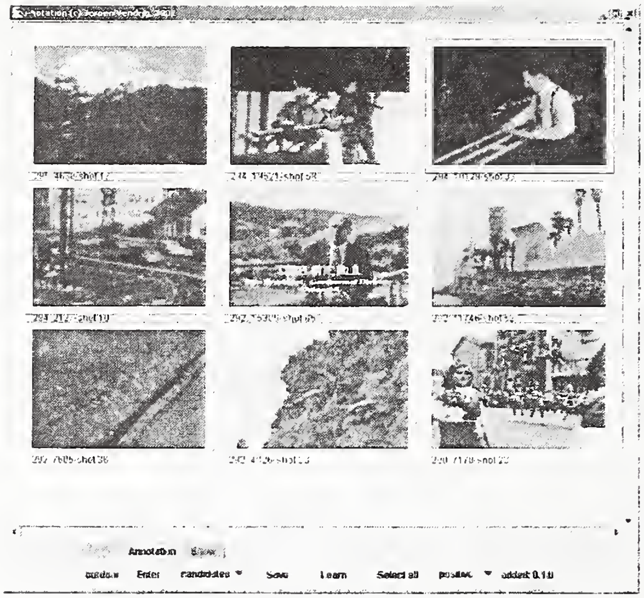


Figure 1: Screendump of the i-Notation system extended with active learning functionality.

6 Results and discussion

The likelihood threshold on the classifier decision confidence has a large impact on the evaluation, as the results lists are smaller than the maximum of 1000 shots for evaluation. The results lists vary from 744 shots for *outdoors* to 4 for *landscape*. The overall TREC results show that many more shots with the same feature are available in the collection, indicating that our threshold is too high.

Employment of a large descriptor pool does not have a negative impact on results according to a comparative experiment. We estimated results for the *face* feature using the "face presence" descriptor only, which is obviously a very powerful descriptor for this feature. Although adding the other descriptors do not lead to better classification results, they do not corrupt the classification either.

7 Conclusions and future research

The use of active learning in combination with the Maximum Entropy classifier leads to a generic approach for feature classification applying to a specific collection. The results of classification for the eight visual TREC features are not satisfactory. This may be due to the heterogeneity of the TREC collection, which is composed of several semantically unrelated collections. Although our approach should be able to cope with such a collection as well, the active learning component should be designed to take the heterogeneity into account. That is, the active learning component should take examples from all sub-collections for labeling by the user, thereby avoiding a local optimum.

The threshold used to determine positive decisions is too high. In the context of TREC evaluation, it would be better to rank all decision confidence values. Further research is needed to find out whether a ranking approach conflicts with the theory on which the Maximum Entropy classifier and its implementation are based.

The use of a large descriptor pool does not confuse the classifier when one very powerful. specific descriptor for a feature is present. as in the case of the *face* feature. Therefore, in future experiments we intend to use more descriptors rather than less. However, an automatic mechanism for selecting relevant descriptors from the pool for a particular feature is desired to be more robust against noise. Although initial experiments on the video data set do not yet show significant effects. we found selection and combination of descriptors to be useful in other Maximum Entropy applications.

The most important future research theme for use of Maximum Entropy classification in multimedia is the binning function. The effect of variations on the current binning function need to be measured. Examples of variations are small versus large amount of bins, determination of amount of bins for each descriptor individually, and using overlapping bins to deal with border values.

Acknowledgements

The authors wish to thank Jan Baan for providing camera work techniques and Thang Viet Pham for the face presence detection.

References

[1] J. Baan. Camera techniek detectie. Technical report, TNO, November 2000.

[2] J. Baan. A. van Ballegooij. J.-M. Geusebroek. D. Hiemstra, J. den Hartog. J. List. C. Snoek, I. Patras, S. Raaijmakers. L. Todoran. J. Vendrig. A. de Vries. T. Westerveld. and M. Worring. Lazy users and automatic video retrieval tools in (the) lowlands. In *Proceedings of the 10th Text Retrieval Conference (TREC)*, pages 104–113. Gaithersburg, USA, November 2001.

[3] J. Baldridge. T. Morton, and G. Bierner. The opennlp maximum entropy package. Technical report, SourceForge, 2002.

[4] A. Berger. S. Della Pietra. and V. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*. 22(1):39–71. 1996.

[5] D.A. Cohn, L. Atlas, and R.E. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221. 1994.

[6] J.N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[7] A. Hanjalic, G.C. Langelaar, P.M.B. van Roosmalen, J. Biemond, and R.L. Lagendijk. *Image and Video Databases: Restoration, Watermarking and Retrieval*, volume 8 of *Advances in Image Communication*. Elsevier, Amsterdam, 2000.

[8] A.K. Jain, R.P.W. Duin, and J. Mao. Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[9] E.T. Jaynes. Information theory and statistical mechanics. *The Phyiscal Review*, 106(4):620–630, 1957.

[10] T. V. Pham, M. Worring, and A. W. M. Smeulders. Face detection by aggregated bayesian network classifiers. *Pattern Recognition Letters*, 23(4):451–461, February 2002.

[11] J.R. Smith, S. Srinivasan, A. Amir, S. Basu, G. Iyengar, C.-Y. Lin, M.R. Naphade, D.B. Ponceleon, and B. Tseng. Integrating features, models, and semantics for trec video retrieval. In *Proceedings of the 10th Text Retrieval Conference (TREC)*, pages 96–103. Gaithersburg, USA. November 2001.

[12] C.G.M. Snoek and M. Worring. Multimodal video indexing: A review of the state-of-the-art. *Multimedia Tools and Applications*. To appear.

[13] R.K. Srihari. Automatic indexing and content-based retrieval of captioned images. *IEEE Computer*, 28(9):49–56, 1995.

[14] H. Sundaram and S.-F. Chang. Determining computable scenes in films and their structures using audio visual memory models. In *Proceedings of the 8th ACM Multimedia Conference*, Los Angeles, CA, 2000.

[15] D.M.J. Tax. *One-class classification*. PhD thesis, TU Delft, 2001.

[16] J. Vendrig and M. Worring. Interactive adaptive movie annotation. *IEEE Multimedia*, To appear.

Microsoft Cambridge at TREC 2002: Filtering track

S E Robertson* S Walker H Zaragoza R Herbrich

Microsoft Research Ltd, 7 J.J.Thomson Avenue, Cambridge CB3 0FB, UK
email {ser, sw, hugoz, rherb}@microsoft.com

1 Summary

Six runs were submitted for the Adaptive Filtering track, four on the adaptive filtering task (ok11af??), and two on the routing task (msPUM?). The adaptive filtering system has been somewhat modified from the one used for TREC–10, largely for efficiency and flexibility reasons; the basic filtering algorithms remain similar to those used in recent TRECs. For the routing task, a completely new system based on perceptrons with uneven margins was used.

2 Okapi at TRECs 1–10

A summary of the contributions to TRECs 1–7 by the Okapi team, first at City University London and then at Microsoft, is presented in [6]. In TRECs 7–10 we took part in the adaptive filtering track, initially concentrating on the thresholding problem, but by TREC–9 we had a full adaptive filtering system with query expansion as well as adaptive thresholding. This adaptation could be used to optimise performance on a number of effectiveness measures and produced good results on both the TREC–9 measures, linear utility and the 'precision-oriented' measure, but performed poorly on the Reuters topics at TREC–10. In earlier TRECs on various adhoc tasks we had concentrated on the weighting schemes and pseudo relevance feedback (blind feedback), and had developed the successful BM25 weighting function but had had only limited success with blind feedback.

3 Adaptive Filtering

3.1 Okapi systems

At the Microsoft Research laboratory in Cambridge, we are developing an evaluation environment for a wide range of information retrieval experiments. This environment is called Keenbow. The various Okapi systems discussed below are seen as components of Keenbow. Many aspects of the systems, including the weighting scheme and the query expansion methods used, reflect the various components of the probabilistic model of retrieval discussed at length in [9].

The Okapi Basic Search System (BSS), which has been used in all Okapi and Okapi/Keenbow TREC experiments up to TREC–9, is a set-oriented ranked output system designed primarily for probabilistic-type retrieval of textual material using inverted indexes. There is a family of built-in weighting functions collectively known as BM25, as described in [5, Section 3] and subsequent TREC papers. In addition to weighting and ranking facilities it has the usual boolean and quasi-boolean (positional) operations and a number of non-standard set operations. Indexes are of a fairly conventional inverted type. Preprocessing includes stopping and stemming and matching a small exceptions dictionary (selected phrases, synonyms and words marked as not suitable for query expansion).

The primary method of using the BSS in adaptive filtering upto TREC–10 was to accumulate small batches of documents, index each batch as a separate BSS database, and search the profiles against it. This was not a very efficient process, and has some limitations – for example, adaptation could only be between batches (according to the TREC filtering track rules). For TREC 2002, we developed a new Okapi/Keenbow component called the Basic Filtering Dogsbody (BFD). The primary principle of the BFD is that a database of profiles (queries) is maintained, and each incoming document is searched against this database. In some sense this makes it a true filtering system, as opposed to an adhoc search system adapted for filtering. The BFD itself does not maintain a cumulative database of documents, but does keep up-to-date the dictionary part of such a cumulative database, consisting of terms and collection frequencies.

Adaptive methods are divided into query expansion or modification and threshold adaptation. Query expansion is performed by the BFD, on the basis of the text query and the cumulated set of known relevant documents (the most recent ones only if there are many). Threshold adaptation is performed by a script built on top of the BFD. This normally involves a search on the reference database, i.e. the cumulative database of all documents received so far. This is a conventional BSS database, and as previously is built in batches (and is therefore not completely up-to- date). Other aspects of the filtering operation, including the history and current state of the profile, are also built as scripts. The master script defines a set of rules for triggering the adaptive procedures: for TREC 2002, the main trigger for updating a profile (query expansion and threshold adaptation) is the retrieval of a relevant

*Also at City University, London, UK.

document. In the experiments described, this happens at every relevant document, and is immediate (i.e. before the next incoming document is processed). The same procedures are triggered occasionally for documents that have failed to retrieve a relevant document for some time.

The adaptive filtering runs were done on a 550MHz Xeon (512KB Cache) with 2Gb RAM and a Dell with two 400 MHz Pentium processors and 512 Mb. Both machines were running Solaris 7. The network was 100Mbps ethernet.

3.2 Algorithms and parameters

Reports from the last two years [7, 8] contain fairly detailed accounts of the filtering system and the adaptation methods used, in particular the relation between the optimisation measures and the threshold. In respect of the algorithms used, this year's system is very similar to last year's; Table 1 is an attempt to summarise the large number of parameters used. Essentially these parameters were set by a series of tuning experiments on the OHSU filtering database (the OHSUMED test collection, adapted for the filtering task for TREC–9, with the OHSU topic set). While this collection is rather different from the Reuters collection, the intention was to look for parameters that would be generally good, rather than ones that would be highly tuned to a particular database. This aim will be furthered by later work on this year's collection, to see how far from optimal the chosen values are. The one parameter which was adjusted from its best value for OHSU was the target number of documents for initial threshold setting. Since this parameter is an absolute number to be retrieved over the entire test set, it is highly dependent on test set size – in fact it would be better expressed as a proportion or probability than as an absolute number. However, on top of this consideration, the OHSU tuning suggested a rather lower value for utility optimisation than for fbeta optimisation.

3.3 Overview of the filtering procedure

At a particular iteration of the process, any query modification needs to take place before any threshold setting. It may also be necessary, after query reformulation but before threshold setting, to recalculate the scores of the previously-retrieved documents, for the adaptation of β.

The document collection is processed a document at a time. If a document is retrieved for any profile, it is immediately checked for relevance. If relevant, the query is updated and then the threshold is updated. At intervals defined by the batch size indicated in the table, the reference database is updated with all documents which have arrived since the last batch. Also, any profile that has not been updated since the last batch is updated.

3.4 Filtering results

As with the official track results, the measures reported are T11SU (scaled utility), T11F (Fbeta measure with beta=0.5), set precision and set recall.

Four runs were submitted, labelled ok11af[ls][ub]. Those with final letter u were optimised for T11SU, and those with final letter f for T11F. The next-to-last letter represents the source of the text topics – l (long) indicates the full text (title, description and narrative), and s (short) denotes title only. In common with other participants, we found very large differences between our performance on the assessor and intersection topics.

The results shown in Table 2 relate to assessor topics only. They are also very slightly different from the official runs, following discovery of a small bug in the system used. Evaluation is based on the full relevance judgements used for the official evaluation. For the runs corresponding to the official runs, adaptation is based on the relevance judgements available for that purpose. Additional runs were made using all relevance data for adaptation. The coding of the runs is:

lms long, medium or short initial topics (medium = title + description)

ub optimised for utility or FBeta

OR adaptation using original or complete relevance judgements

Disappointingly, the runs optimised for utility do marginally better on the FBeta measure than the run optimised for FBeta, at least when using the original relevance data. (This is the exact opposite of the result for last year!). It seems that the method for setting thresholds for FBeta, which involves estimating the total relevant in the collection, is producing somewhat erratic results. Further diagnostic testing is required.

Starting with longer topics may help a little (on utility at least) but the differences do not seem consistent (medium length topics seem to have no advantage over short ones). It seems from the assessor topic results at least that it is possible for an adaptive filtering system to bootstrap its performance reasonably well even if the starting point is not very good.

Intersection topics

However, it is difficult to reconcile the tentative conclusion above with the terrible performance on the intersection topics. One possible suggestion is that the 'relevance' judgements for the intersection topics (i.e. the assignment of documents to two different topic codes by Reuters editorial staff) fail to define a set of related documents with the sort of coherence that we find in assessor relevance judgements. Another is that the pairs of topics may have been unbalanced in some way, leaving it difficult for the filtering systems to infer criteria covering both aspects.

For the run corresponding to ok11aflu, the results are: T11SU=0.251, T11F=0.040, Precision=6.6%, Recall=2.3%. All the others are similarly bad or worse. We looked in detail at two topics, R195 and R181. R195 is formed by the intersection of Reuters topic categories GVOTE (Elections)

Table 1: Parameters for adaptive filtering

See notes below and [7, 8] for explanations of these parameters	
BM25 parameters:	
k_1	1.3
b	0.55
Score calibration:	
These parameters define the mapping from Okapi score to probability of relevance – $P(R)$ is estimated as a linear function of score, slope Gamma and intercept Beta. At each threshold updating, Beta (but not Gamma) is re-calibrated using scores of documents of known relevance. The 'mythical reldocs' serve as a Bayesian prior in this re-calibration.	
Initial beta	-0.66
Mythical reldocs for beta re-calibration	3
Gamma	2.9
Threshold adaptation:	
Initially, the threshold is set at a level estimated to retrieve a certain target number of documents over the whole test set. As relevant documents are retrieved, the threshold is moved up a ladder until it reaches the level defined by optimising the required parameter.	
Initial target no. of documents (FBeta)	70
Initial target no. of documents (Utility)	25
Ladder step	2
Query modification:	
Query modification uses the last n relevant documents retrieved (including the training sample if necessary), together with the original text query. Terms are ranked by absolute term selection value (new offer weight). All those exceeding the threshold are chosen, subject to both a minimum and a maximum number of terms.	
Reldocs used for modification	20
Maximum terms	25
Minimum terms	3
Absolute term selection value threshold	2
Document batching:	
Determines how often the accumulated reference database is updated, and also how often the threshold updating procedure is initiated for profiles which have retrieved no relevant documents since the last such update.	
Batch size	50,000
Further notes on thresholding For Utility, the threshold calibrated as a log-odds probability is raised by one ladder-step for each relevant document retrieved. This is then compared with the level defined by the utility function, and the lower of the two is chosen. After 8 relevant documents have been retrieved, the level defined by the utility function is always chosen. For Fbeta, a similar procedure is followed, but instead of the level defined by the utility function, the estimated optimum Fbeta threshold is used. The ladder function is different from last year. The target is reduced pro-rata according to the estimated remaining number of documents to come, and then further divided by $((ladder\ step)**(numberofrelevantdocuments))$. Thus if the ladder step is set to 1, the ladder is effectively switched off. Higher values give larger steps.	

Table 2: Main results

Topics	Relevance judgements used for adaptation	Corresponding official run	T11SU	T11F	Precision	Recall
Utility optimisation						
long	original	ok11aflu	0.435	0.421	49.9	34.4
long	all		0.439	0.419	46.8	37.8
medium	original		0.405	0.405	48.5	31.9
medium	all		0.412	0.405	46.8	34.4
short	original	ok11afsu	0.406	0.404	48.2	33.0
short	all		0.418	0.413	46.2	36.7
FBeta optimisation						
long	original	ok11aflb	0.405	0.394	52.4	26.1
long	all		0.410	0.405	50.4	28.4
medium	original		0.396	0.392	52.0	26.3
medium	all		0.411	0.415	50.6	29.7
short	original	ok11afsb	0.404	0.393	52.0	25.9
short	all		0.418	0.411	50.8	29.0

Table 3: Titles of relevant and retrieved documents, topic R195

Training relevant:	1	Churches put poverty on NZ election agenda
	2	Dole accuses Clinton of “mediscare” ad campaign
	3	Clinton blocks federal loans to deadbeat parents
Test relevant:	4	Florida’s elderly key to Dole campaign
	5	U.S. group seeks child food-aid support
	6	Poverty is toughest task for next Nicaraguan leader
	7	Dole visits Florida, promises to save medicare
	8	Relaxed, confident Clinton stumps in central Florida
	9	Clinton would mull law aiding retirees if elected
	10	Arizona voters back lottery measure
	11	NZ’s National, Labour agree to pension referendum
	12	Poland’s pension reform under election cloud
	13	UK’s Dorrell details old age care insurance plan
	14	UK welfare reform to head Major’s election agenda
	15	Polish Solidarity sees growth as top economic goal
Retrieved:	16	S. Africa releases conservative welfare blueprint
	17	NYC agency says welfare poses big budget challenge
	18	The inexorable GST [Australian sales tax]
	19	New Moldovan leader seen backing market reforms
	20	Despite good times, many in U.S. need charity
	21	HUD chief warns U.S. near housing crisis for poor
	22	Study finds up to 10% of Swiss are poor
	23	UK’s Blair to unveil welfare plans
	24	British magazine offers help to homeless
	25	French government approves anti-poverty plan
	26	UK Labour’s Brown vows no tax and spend cure-all
	27	French MPs debate controversial anti-poverty bill

Table 4: Titles of training relevant documents, topic R181

Training relevant:	1	FoxMeyer Drug declares bankruptcy after sale falls through
	2	Foxmeyer says drug unit files for bankruptcy
	3	Westa receiver seeks Prochnik manager

and GWELF (Welfare, Social Services); R181 from C16 (Insolvency/liquidity) and C411 (Management Moves). In both these cases, as in many other intersection topics, there is no overlap at all between test relevant and retrieved: recall, precision, FBeta, unnormalised utility are all zero.

For topic R195, titles of the 3 training documents for adaptive filtering are given in Table 3, together with most of the relevant documents from the test set, and most of those retrieved in run ok11aflu (a few, including some duplicates, have been left out in the interests of saving space).

It may be seen that the documents found by the system are broadly in the right area – some look less obviously good candidates than others, but there are several in the list which one might reasonably expect to be relevant. One issue is that it seems that in order to qualify for the Election & Welfare category in Reuters, a document has to relate to a particular election. This probably excludes some of the retrieved documents, but not for example number 23, which does indeed relate to the impending British general election, exactly as do 13 and 14. However, 23 was assigned (in addition to GWELF) the code GPOL (Domestic Politics) but not GVOTE. One can only conclude that in this instance at least, the Reuters coding is just not very consistent. Number 26 is even worse – it has various headings relating to economics and finance, and GPOL and GCAT (Government/Social), but not GVOTE (despite the fact that it reports a campaign speech by someone not then in government) and not GWELF (despite the fact that a significant part is about poverty and unemployment).

We might have hoped to retrieve at least some of the relevant set. However, the filtering system is quite sensitive to adaptation – if it is getting no encouragement (in the form of positive relevance judgements) it will keep the threshold very high (the penalties for allowing through much more are too great).

In the case of topic R181, we show just the three training examples in Table 4

In this case, two of the titles relate to the same story. The interpretation of Reuters topic C411 (Management Moves) is supposed to be moves such as management appointments or resignations. Number 1 has a brief mention of an appointment in a story about the bankruptcy of FoxMeyer; number 2 is essentially an abbreviated version of no. 1, though the appointment part has been retained. Number 3 has (in our interpretation) no management moves in the sense given at all: the receiver is seeking not an individual but a financial institution to manage and sell a stake in another company. The ok11aflu run retrieved 12 documents, all squarely in the insolvency area, but none containing management moves. (Several of them relate to FoxMeyer, but there is also a group relating to Bulgarian banks. The one Bulgarian bank story which was marked as relevant was not selected in ok11aflu.) Thus this example seems to be an instance of one of the two original Reuters categories dominating. However, part of the reason is the choice of positive examples for training – it is certainly the case that those particular examples emphasise only one of the Reuters categories.

Reuters categories are often very broad concepts, and must be hard to assign consistently. On the evidence of these two cases, one might suggest that the intersection operation, together with the accidental choice of training examples, has significantly compounded the noise.

4 Routing

The perceptron-based system was developed for the TREC routing task independently of Okapi. The theoretical work leading to this model was carried out in 2001 and first evaluations on smaller datasets (such as Reuters-21578) were carried out at the beginning of this year [4]. Our TREC 2002 runs constituted the first full-scale implementation and evaluation of this model.

Research on Perceptrons is motivated by the recent success of soft-margin support vector machines for routing [3]. Soft-margin support vector machines are high-dimensional linear classifiers that maximise a quantity called the *margin* while keeping the training error close to zero. Because of the intimate relationship between margin and generalisation error, maximising the former will (asymptotically) minimise the latter.

When the training set is not linearly separable in its feature space the margin is maximised while allowing a small number of misclassification errors. The *cost* of a misclassification is determined prior to training by a learning parameter. C. An additional parameter, j, is used to weight differently positive and negative misclassifications. These two parameters are set in general by k-fold cross-validation ([3]).

Different theoretical and practical reasons made us search for alternative solutions to the SVM for the task of document routing:

1. It is theoretically not clear under which conditions large margin classifiers may lead to good *rankings* (as opposed to good classification).

2. There are other linear classifiers which do not maximise the margin but perform as well as the SVM for many classification tasks. Generalisation error bounds for these algorithms exist and some are tighter than those of the soft margin SVM.

3. Training times for SVMs are extremely long.

4. The need to optimise C and j multiplies the number of times we need to train the systems.

In particular, the perceptron learning algorithm (PLA) is a fast learning algorithm for linear classifiers, and it has been shown recently that it shares with the SVM some strong theoretical properties. In particular. one can show that *sparsity* for the perceptron (roughly speaking, the number of training updates) works similarly to margin for the SVM. that is. high sparsity guarantees low generalisation error. Furthermore. it has been shown that the existence of a large margin solution implies the high sparsity of perceptron solutions. This means, again roughly speaking, that if there exists a good SVM solution (that is, one with a large margin) then the perceptron solution on the same dataset is likely to be good as well (see [1] [4] for a more formal discussion of these topics).

Our initial experiments in routing with the PLA (using the Reuters-21578 topics collection, and the average precision performance measure) showed that although it was slightly outperforming the SVM for topics with many positive examples, it underperformed significantly for smaller topics. This seems to indicate that one needs to impose some margin constraints on very small topics.

The margin-PLA [2] is a modified PLA which guarantees a solution with a minimum margin, i.e. the resulting margin is within a factor of $\tau/(2\tau+1)$ of the maximum possible margin (which would be found by an SVM). τ is therefore a parameter (similar to C) which must be set prior to training. While experiments with the margin-PLA showed improvement in performance over the PLA for small topics, it greatly increased the training time and decreased the sparsity of the solution. One of the reasons for this is that the margin constraints are symmetrical, that is, if we wish to enforce a large margin with respect to the relevant documents, we must do the same with respect to the irrelevant documents — a task that is too expensive because of their large number.

For these reasons, we modified the margin-PLA algorithm to take account of the asymmetry of the problem, and we replaced the constant τ by two constants, τ_{+1} and τ_{-1}, which enforce different margins with respect to the relevant (+1) and irrelevant (−1) documents. This led to a great improvement of the speed of the training algorithm and the sparsity of the resulting solutions. Furthermore, when we optimised by cross validation the parameters τ_{+1} and τ_{-1} the resulting solutions outperformed the SVM on Reuters-21578 [4].

In the following sections we describe our algorithm, the perceptron learning algorithm with uneven margins (or PLAUM), its implementation for the TREC 2002 routing task, and summarise the results obtained.

4.1 The PLAUM algorithm

We present in Algorithm 1 the PLAUM as implemented for our TREC 2002 routing experiments. Basically, we iterate over the training sample testing for every pattern $\vec{x}_i$ if the output of our classifier $(\langle\vec{w},\vec{x}_i\rangle + b)$ is of the right sign and, even more, greater than the required factor on the minimal margin for the pattern's class y_i, τ_{y_i}. When *all* the patterns satisfy this condition, the algorithm stops.

Despite the high dimension of documents (from hundreds to tens of thousands) linear separability cannot always be guaranteed. This condition can be relaxed by the so-called λ-*trick*, which extends each document vector $\vec{x}_i$ by a vector of size m with value λ for the ith coordinate and zero elsewhere (m is the number of training documents). To implement this it suffices to redefine the inner-product function as: $\langle\vec{w},\vec{x}_i\rangle := \sum_{j=1}^{m} w_j x_{i,j} + w_{m+i}\lambda$.

The PLAUM algorithm with the λ-trick is guaranteed to always stop at a solution if $\lambda > 0$. Nevertheless, in some pathological cases the algorithm can iterate a very large number of times. For this reason we include the parameter T which sets a maximum to the number of epochs (iterations over the training set) allowed.

Finally, for completeness we have included in the algorithm the learning parameter η. However, in our experiments this parameter was always set to 1.

Algorithm 1 PAUM $(\tau_{-1}, \tau_{+1}, T, \eta, z)$

```
Require: A linearly separable training sample
  $z := (x, y) \in (\mathcal{X} \times \{-1, +1\})^m$
Require: A learning rate $\eta \in \mathbb{R}^+$
Require: A maximum epochs parameter $T$
Require: Two margin parameters $\tau_{-1}$, $\tau_{+1} \in \mathbb{R}^+$
  epoch ← 0; $i$ ← 1; updated ← $m$
  $\vec{w} \leftarrow \vec{0}$; $b = 0$; $R \leftarrow \max_{\vec{x}_i \in x} ||\vec{x}_i||$
  repeat
    if $y_i\,((\langle\vec{w},\vec{x}_i\rangle + b) \leq \tau_{y_i}$ then
      $\vec{w} \leftarrow \vec{w} + \eta y_i \vec{x}_i$
      $b \leftarrow b + \eta y_i R^2$
      updated ← $i$
    end if
    $i \leftarrow i + 1$
    if $(i > m)$ then
      $i \leftarrow 1$; epoch ← epoch+1
    end if
  until ($i$ = updated) or (epoch $\geq T$)
  return (w, $b$)
```

4.2 Data representation

We considered two different representations of the documents: the usual *tf*×*idf* representation and a BM25-based representation where *idf*'s are replaced by topic-dependent BM25 weights.

Pre-processing was kept to a minimum: no stemming was used nor were stop words removed. Punctuation marks and letter case were removed, and all character strings appearing in fewer than three documents were eliminated. All other character strings became features (terms) of the linear classifier.

For the *tf*×*idf* representation all resulting features in the training set were considered (approximately 10 000). For the BM25-based representation only features in relevant documents were considered (approx. 600 on average). Finally all vectors were normalised to have unit Euclidean norm.

4.3 Model Selection

Two parameters need to be set prior to training: τ_{+1} and τ_{-1}. To choose these values we proceeded as follows:

First, the training set was randomly split into two halves, one half used for training and the other used for testing. Sec-

Table 5: (Submitted Runs) Routing results, PLAUM algorithm. Macro-Average Precision.

Run	TOPICS	MAP	MAP(> .1)
msPUMb	R101-151	0.355	.368 (#48)
msPUMs	R101-150	0.239	.348 (#34)
msPUMb	R151-200	≈ 0	-
msPUMs	R151-200	≈ 0	-

Table 6: (Post-Submission) Effects of τ and Model Selection (see text for details). Macro-Average Precision for all topics (Test) and for topics R101-150 (Train/Test[50]).

Model	Test	Test[50]	Train[50]
PLA	0.211	0.376	0.4801
PLAUM (+1,0)	0.219	0.385	0.513
PLAUM(*)	0.224	0.403	0.54

ond, the 100 models corresponding to the 100 topics were trained independently for $\tau_{+1} \in \{0, 1, 10, 100\}$ and $\tau_{-1} \in \{-10, -1, 0, 1\}$, leading to 16 different runs per topic. This procedure was repeated 5 times, choosing a different random train/test split every time, and performance on different splits was averaged. This resulted in an average precision reading per topic and per (τ_{+1}, τ_{-1}) setting. Finally, for each topic the best (τ_{+1}, τ_{-1}) parameters were selected and used to train the final model over the entire training set.

The training algorithm was run on a 2.5GHz CPU machine with 500Mb of memory. Data was accessed from a SQL server over a 100Mhz Ethernet network. The entire model selection procedure for the 100 topics and 5 splits runs under 5 hours. We believe that code properly optimised for speed could finalise this task under one hour.

4.4 Results

Due to time and resource limitations we restricted our preliminary experiments to the Reuters-21587 routing task, we have not performed any TREC runs besides those submitted.

Two runs were submitted, varying only in the size of feature set used (as discussed in section 4.2), very large for *msPUMb* and small for *msPUMs*. Results are summarised in 4.4.

The large feature set model msPUMb greatly outperformed the small feature set model on average. This is not surprising, especially when we consider i) how little pre-processing was done with the documents, and ii) the simplicity of the term selection procedure. Nevertheless, for a number of topics the small feature set was better than or similar to the larger feature set. On the left-most column of Table 4.4 we show macro average precision when we average only over the topics obtaining more than 0.1 average precision (we indicated in parenthesis the number of these topics). This figure is very close for both systems, indicating that msPUMs is in fact performing similarly to msPMs for many topics, but it completely underperforms for others. If we could detect such topics at learning time we could adapt the size of the feature sets to the nature of the topics. We are currently working on this problem.

Table 7: (Post-Submission) Some figures of merit of the PLA and the selected PLAUM(*), averaged over all 100 topics.

	PLA	PLAUM(*)
Average Precision	.211	.224
Non-Zero Weights	1179	2236
Epochs	3.6	13.1
Updates	17.5	77.8
Selection time	-	1.87 s.
Train time	.22 s.	-
Train+Test+Submit time	45s.	45s.

There are many algorithms for feature selection and projection which we could have used. However, it has been observed empirically by several authors that using linear classifiers for text seems to benefit from the maximum number of features available. In the absence of space, memory or computational time limitations, we did away with feature selection methods. However, in real operational settings the situation is very different. As one increases the number of features (or similarly if the sparsity of a classifier decreases), the number of potentially relevant documents that need to be scored for each topic increases rapidly. This is very dangerous for systems that must filter simultaneously a large number of topics and documents. Is it then justified to use 10 000 features if 80% of the performance can be obtained using only 50 features? This difficult issue is not addressed by the present TREC evaluation measures.

In tables 4.4 and 4.4 we present some results to demonstrate the superiority of the PLAUM algorithm with respect to PLA and the interest in running a model selection procedure such as the one outlined in this paper. For these comparisons we consider only the msPUMb model. We note that these results are better than those submitted originally: after submission we discovered an error in our data normalisation procedure; after correcting it the performance of all models was increased.

In table 4.4 we compare macro-average precision performance (for all topics and for only the first 50) of the original PLA algorithm, a simple PLAUM model with $(\tau_{+1} = +1, , \tau_{-1} = 0)$, and the PLAUM model obtained using the model selection procedure discussed in 4.3. We observe that the original PLA algorithm yields very good performance already and that enforcing some positive margin (i.e. $\tau_{+1} = +1$) increases this performance further. Nevertheless, the best results are obtained when the τs are selected for each topic.

In table 4.4 we compare several figures of merit of the original PLA and our PLAUM(*) model. As expected, learning the PLAUM(*) model requires more updates and more epochs, but its sparsity is not greatly reduced and the resulting training and testing times are perfectly reasonable. In fact, once the model selection step is completed, the difference in training time is negligible compared to IO and scoring time.

5 Conclusions

The performance of the basic Okapi filtering system, tuned for OHSUMED data but run on this year's Reuters task, is fair but not outstanding. The problem of estimating the total number of relevant documents in the entire collection, which is necessary for optimising the FBeta measure, has not been investigated further since last year; it may be one reason why the FBeta-optimised runs performed worse on FBeta than the utility-optimised runs.

The PAUM method for routing appears promising. It could be applied to batch filtering (we have not yet done so); but as with many such machine learning methods, it presents problems if we want to apply it to adaptive filtering. This remains a challenge.

Our performance (with two very different methods on two different tasks) on the intersection topics was extremely poor. This may be because they are simply more difficult, but we suspect that the intersection method is not a very good way to define sufficiently coherent topics.

References

[1] Thore Graepel, Ralf Herbrich, and Robert C. Williamson. From margin to sparsity. In *Advances in Neural Information Processing Systems 13*, pages 210–216, 2001.

[2] W. Krauth and M. Mézard. Learning algorithms with optimal stability in neural networks. *Journal of Physics A*, 20:745–752, 1987.

[3] David Lewis. Applying support vector machines to the trec-2001 batch filtering and routing tasks. In *Text Retrieval Conference (TREC-10)*, pages 286–292, 2001.

[4] Yaoyong Li, Hugo Zaragoza, Ralf Herbrich, John Shawe-Taylor, and Jaz Kandola. The perceptron algorithm with uneven margins. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Nineteenth International Conference on Machine Learning, ICML'2002*, Cambridge, MA, 2002. MIT Press.

[5] S E Robertson et al. Okapi at TREC–3. In D K Harman, editor, *Overview of the Third Text REtrieval Conference (TREC–3)*, pages 109–126. Gaithersburg, MD: NIST, 1995. NIST Special Publication 500-225.

[6] S E Robertson and S Walker. Okapi/Keenbow at TREC–8. In E M Voorhees and D K Harman, editors, *The Eighth Text REtrieval Conference (TREC–8)*, pages 151–162. Gaithersburg, MD: NIST, 2000. NIST Special Publication 500-246.

[7] S E Robertson and S Walker. Microsoft Cambridge at TREC-9: Filtering track. In E M Voorhees and D K Harman, editors, *The Ninth Text REtrieval Conference (TREC–9)*, pages 361–368. Gaithersburg, MD: NIST, 2001. NIST Special Publication 500-249.

[8] S E Robertson, S Walker, and H Zaragoza. Microsoft Cambridge at TREC-10: Filtering and web tracks. In E M Voorhees and D K Harman, editors, *The Tenth Text REtrieval Conference, TREC 2001*, pages 378–383. Gaithersburg, MD: NIST, 2002. NIST Special Publication 500-250.

[9] K Sparck Jones, S Walker, and S E Robertson. A probabilistic model of information retrieval: development and comparative experiments. *Information Processing and Management*, 36:779–808 (Part 1) and 809–840 (Part 2), 2000.

Extracting Answers from the Web Using Knowledge Annotation and Knowledge Mining Techniques

Jimmy Lin Aaron Fernandes Boris Katz Gregory Marton Stefanie Tellex
MIT Artificial Intelligence Laboratory
200 Technology Square
Cambridge, MA 02139
{jimmylin,adfernan,boris,gremio,stefie10}@ai.mit.edu

Abstract

Aranea is a question answering system that extracts answers from the World Wide Web using knowledge annotation and knowledge mining techniques. Knowledge annotation, which utilizes semistructured database techniques, is effective for answering large classes of commonly occurring questions. Knowledge mining, which utilizes statistical techniques, can leverage the massive amounts of data available on the Web to overcome many natural language processing challenges. Aranea integrates these two different paradigms of question answering into a single framework. For the TREC evaluation, we also explored the problem of answer projection, or finding supporting documents for our Web-derived answers from the AQUAINT corpus.

1 Introduction

Aranea, MIT's entry to the TREC Question Answering track, focused on extracting answers from the World Wide Web. Our system was organized around a modular framework that integrates two different paradigms of question answering: knowledge annotation using annotated structured and semistructured resources and knowledge mining using statistical techniques that leverage data redundancy (cf. (Lin and Katz, 2003)).

Aranea's approach to question answering is motivated by an observation about the empirical distribution of user queries, which quantitatively obey Zipf's Law—a small fraction of question types account for a significant portion of all question instances. Certain natural language questions tend to occur much more frequently than others, an observation that is confirmed by many different sources: TREC queries (Lin, 2002), logs from the START question answering system (Katz, 1997), and logs of a commercial search engine (Lowe, 2000). Furthermore, many questions ask for the same type of information and differ only in the specific object questioned, e.g., "What is the population of (the United States, Mexico, Canada,...)?" We can group these questions into a single class (or type), i.e., "What is the population of x?" where x can be any country, and find the answer in a database. *Knowledge annotation* is a question answering strategy that allows heterogeneous sources on the Web to be accessed as if it were a uniform database. By connecting natural language queries to this "virtual" database, Aranea can answer large classes of commonly-occurring questions.

Typically, Zipf curves have broad tails where individual instances are either unique or account for an insignificant fraction of total instances. This observation also holds true for the distribution of user questions: in addition to asking large classes of commonly-occurring questions, users also pose a significant number of unique questions that cannot be easily classified into common categories or grouped by simple patterns, e.g., "What format was VHS's main competition?" To answer these questions, Aranea employs what we call redundancy-based *knowledge mining* techniques.

For the TREC evaluation, the extraction of answers from the Web necessitated an extra step in the question answering process, usually known as *answer projection*. For every Web-derived answer, our system had to find a supporting document from the AQUAINT corpus, even though the corpus itself was never used in the question answering process. This additional component had a significant impact on the overall performance of our system.

2 Overall Framework

The general architecture of the Aranea system is shown in Figure 1. User questions are routed to two separate components, one that employs the knowledge annotation strategy (Section 3) and one that utilizes the knowledge mining strategy (Section 4). Both components consult the World Wide Web to generate candidate answers, and the results of both components are piped through a knowledge boosting and cleanup module (Section 5), which check the answer candidates against a number of heuristics to ensure their validity. Finally, the answer projection module (Section 6) finds an article from the AQUAINT corpus that adequately supports the answer derived from the Web.

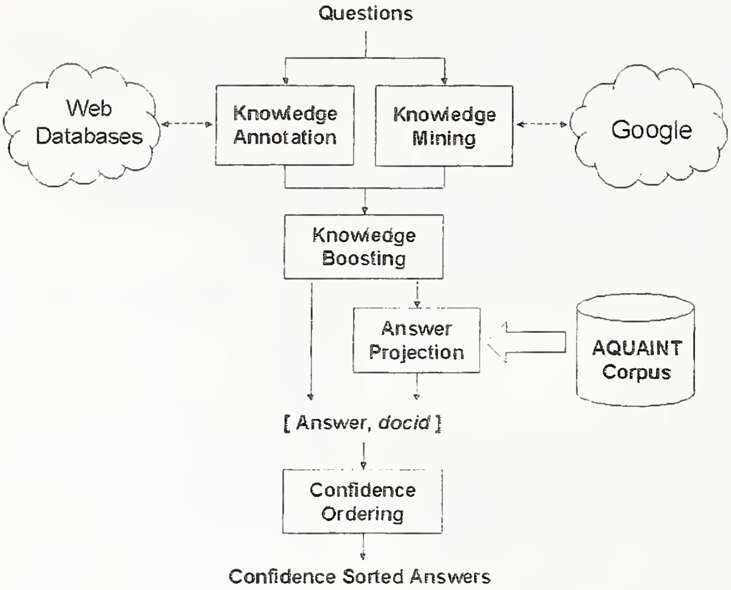


Figure 1: Overall Architecture of the Aranea question answering system.

Aranea supports a modular pipeline architecture by enforcing input and output constraints at the module interfaces. The input and output of each module is an XML-encoded data structure that keeps track of the current computational state. Aranea modules are conceptualized as transformations over this XML data structure.

3 Knowledge Annotation

Although the Web consists largely of unorganized pages, pockets of structured and semistructured knowledge exist as valuable resources for question answering. For example, the CIA World Factbook provides political, geographic, and economic information about every country in the world; 50states.com contains numerous properties related to US states from state bird to land area; Biography.com has collected profiles of over twenty-five thousand famous (and not-so-famous) people; the Internet Movie Database stores entries for hundreds of thousands of movies, including information about their cast, production staff, and dozens of other properties.

To effectively use these existing resources for question answering, the plethora of knowledge sources must be integrated, or federated, under a common interface or query language. Database concepts and techniques provide the tools to accomplish just that. In fact, since many of these sources are part of the "deep" or "invisible" Web, they are inaccessible to search engines and can only be modeled as "virtual" databases. We have developed a schema-based technique called knowledge annotation by which natural language queries can be connected to semistructured knowledge sources.

Our knowledge annotation strategy provides an effective mechanism for answering natural language questions. Because users frequently ask the same types of questions, a few well-chosen knowledge sources are sufficient to provide good knowledge coverage. For example, we have verified that ten Web sources can provide answers to 27% of TREC-9 and 47% of TREC-2001 questions from the QA track (Lin, 2002). In addition, other researchers (Hovy et al., 2002) have noticed the importance of external knowledge sources for question answering.

The knowledge annotation component of Aranea is a simplified implementation of the system used by the START (Katz, 1988; Katz, 1997) and Omnibase (Katz et al., 2002a; Katz et al., 2002b) systems. START is a natural language understanding system, and Omnibase is a virtual database that provides uniform access to heterogenous and distributed Web sources via a wrapper-based framework. A simplified version of natural language annotation technology (Katz, 1997) is employed in database access schemata to mediate between natural language and database queries.

Since it came on-line in December 1993, START has engaged in exchanges with hundreds of thousands of users all over the world, supplying them with useful knowledge. However, because the system provides users with paragraph-sized answers that often contain multimedia fragments such as pictures and audio clips, they are not suitable for a TREC-style evaluation. There is evidence to support that

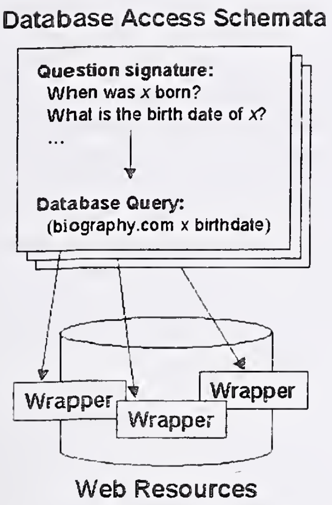


Figure 2: The knowledge annotation component of Aranea

paragraph-sized chunks form the most suitable unit of response to a user question, because it provides not only the exact answer, but additional contextual information that may help with interpretation and analysis (Lin et al., 2003). Because this year's TREC QA track accepted only exact answers, we found it inappropriate to directly evaluate START and Omnibase.

3.1 Database Access Schemata

A collection of database access schemata and wrappers comprise the knowledge annotation component of Aranea (Figure 2). Each schema is composed of two connected parts: the question signature and the database query. A question signature is a collection of regular expressions that match a specific class of user questions, e.g., requests for birth dates of people.[1] These patterns are paired with unfilled database queries that are dynamically instantiated with bindings extracted from the question signature. Consider a typical database access schema:

When was x born?
What is the birthdate of x?
...
→ (biography.com x birthdate)

In this example, questions that ask for the birth dates of various people are translated into an *object-property-value* database query (Katz et al., 2002a). These queries specify the data source where the answer could be found (biography.com), the *object* in question (x), and the *property* sought after (birthdate). The *value* of the object's property typically answers the user's question.

The knowledge annotation component of Aranea operates by matching the user question against schemata stored in the knowledge base and executing database queries generated by the matched schemata.

The Aranea database engine is responsible for retrieving the actual answers by executing the database queries. The retrieval of such information depends on the type of the data source: Some sources are stored locally, and may translate into a file lookup. Other sources are stored on remote Web sites behind a CGI interface; executing database queries on these sources requires dynamically reconstructing an HTTP request and properly parsing the resulting HTML document. More details on the process of structuring a knowledge source for database access is discussed in the next section.

3.2 Knowledge Engineering

Teaching Aranea's knowledge annotation component to answer different classes of natural language questions involves three separate steps borrowed from START and Omnibase: identifying question classes and knowledge sources, writing the database access schemata, and writing wrappers for the data sources.

The first step in the knowledge engineering process is to identify the class of questions to be answered and an adequate knowledge source that provides the answer. Empirical analysis of the question distribution provides hints on the effectiveness of any effort. We have noticed, for example, that users frequently asked about the demographics and economics of countries. These questions can be answered by writing a schema that uses information found within the CIA World Factbook.

Once a question class and a knowledge source have been determined, regular expression patterns that capture the general form of the question must be written. Usually, such patterns take into account various alternative formulations of the same query. These patterns must clearly indicate the noun phrase that can be parameterized. For example, in the question class "What is the GDP of x?", x stands as a generic placeholder for country names. The mapping between the natural language patterns and the database query must also be specified, e.g., the x extracted from the previous question pattern fills the x slot in the database query (cia-factbook x gdp)

After a database access schema has been crafted, a wrapper must be written for the knowledge resource. These wrappers supply the actual procedures used to execute queries of a specific form. Although

[1] These question signatures are a simplified version of natural language annotations used by START, which are parsed into and stored as ternary expressions. Because matching occurs at the level of the parsed structures, powerful linguistic machinery can be employed to handle different linguistic phenomena, e.g., synonymy, hyper/hyponymy, syntactic alternations, etc.

Aranea provides a generic framework for organizing the queries and convenient libraries of often-used functionality, specific implementations for accessing various data sources must be provided separately. Typically, executing a query involves either looking up the information in a locally stored database (ranging in complexity from tab-delimited flat files to full SQL databases), or executing a CGI request to retrieve a dynamically generated page from a remote Website and performing additional postprocessing to extract only the relevant fragments.

The Aranea system deployed for the TREC competition included twenty-eight schemata that access seven different data sources. Here are two examples:

- `Biography.com` This source provides information about the lifespan, birth dates, and death dates of various well-known people. Answering questions about such properties involves dynamically retrieving pages from `biography.com` (via CGI) and performing simple pattern matching on the HTML document to extract exact dates.
- `CIA World Factbook` This resource provides various useful facts about countries, e.g., population, area, capital, etc. This information was download and structured in a locally-stored tab-delimited file. Questions about various properties of world countries are translated into simple file lookups.

4 Knowledge Mining

The knowledge mining approach to question answering is based on the observation that as the size of a text collection increases, occurrences of the correct answer tend to also increases. Specifically, Breck *et al.* (Breck et al., 2001) noticed a correlation between the number of times an answer appeared in the TREC corpus and the average performance of TREC systems on that particular question. This result verifies intuition: the more times an answer appears, the easier it is to find it. The knowledge mining component of Aranea extends this insight to the World Wide Web, and leverages the Web's massive size for question answering.

As a text collection, the Web is larger in size than any research corpus by several orders of magnitude. An important implication of this size is the amount of data redundancy inherent in the text collection; potentially, each item of information has been stated in a variety of ways, in different documents. However, data redundancy is counterbalanced by the poor quality of individual documents.

A question answering system can utilize massive amounts of Web data in two ways: as a surrogate for sophisticated natural language techniques and as a method for overcoming poor document quality. Consider the question "When did Wilt Chamberlain score 100 points?" Here are two possible answers:

> (1) Wilt Chamberlain scored 100 points on March 2, 1962 against the New Yorks Knicks.
>
> (2) On December 8, 1961, Wilt Chamberlain scored 78 points in a triple overtime game. It was a new NBA record, but Warriors coach Frank McGuire didn't expect it to last long, saying, "He'll get 100 points someday." McGuire's prediction came true just a few months later in a game against the New York Knicks on March 2.

The answer could be more easily extracted from sentence (1) than from passage (2). In general, the task of answering a question is not very difficult if the document collection contains the answer stated as a simple reformulation of the question. In these cases, simple techniques, e.g., keywords or regular expressions suffice to achieve state-of-the-art performance. As the size of the document collection grows, the more likely it is that question answering systems can find statements that answer the question by matching a simple reformulation.

Without the luxury of massive amounts of data, a question answering system may be forced to extract answers from passages in which they are not obviously stated, e.g., passage (2). In these cases, sophisticated natural language processing may be required to relate the answer to the question, e.g., recognizing syntactic alternations, resolving anaphora, making commonsense inferences, performing relative date calculations, etc.

The World Wide Web is so big that simple pattern matching techniques can often replace the need to understand both the structure and meaning of language. The answer to a question could be extracted by searching directly for an anticipated answer form. e.g., in the above example, by searching for the string "Wilt Chamberlain scored 100 points on" and extracting words occurring to the right. Naturally, this simple technique depends crucially on the corpus having an answer formulated in a specific way. Thus, the larger the text collection is, the greater the probability that simple pattern matching techniques will yield the correct answer. Data redundancy enables a simple trick to overcome many troublesome issues in natural language processing, e.g., alternations, anaphora, etc.

Despite the apparent advantages of massive amounts of data, the process of answering questions using the Web is complicated by the low average quality of individual documents. Due to the low barrier of entry in Web publishing, many documents are poorly written, barely edited, or simply contain incorrect information. As a result, text extracted from a single document cannot be trusted as the correct answer. This problem can also be alleviated through data redundancy. A single instance of a candidate answer may not provide sufficient justification, but multiple occurrences of the same answer in different documents lends credibility to the proposed answer.[2]

The tremendous amounts of information on the World Wide Web would be useless without an effective method of data access. Providing the basic infrastructure for indexing and retrieving text at such scales is a tremendous engineering task. Fortunately, such services already exist, in the form of search engines. For example, Google, the largest of the Web search engines, boasts over 3 billion documents in its index.[3] Using existing search engines as information retrieval backends, we can focus our efforts on answer extraction.

Many of the knowledge mining techniques described above have been implemented in previous systems (Brill et al., 2001; Buchholz, 2001; Clarke et al., 2001; Kwok et al., 2001; Soubbotin and Soubbotin, 2001; Brill et al.. 2002). The introduction of redundancy-based question answering using the Web (Brill et al., 2001) at last year's TREC conference has generated a new set of techniques for attacking the question answering problem. We have taken advantage of previous experiences to refine many techniques within a better engineered framework. In particular, our infrastructure supports a modular architecture that allows specific functionality to be encoded into manageable components. This not only allows for faster development cycles, but facilitates glass-box testing to properly determine the effectiveness of various techniques.

The data flow in the knowledge mining component of Aranea is shown in Figure 3. In the following sections, we describe each module in detail. Each module accepts an Aranea XML data structure as input and returns a structure of the same type as output; the functionality of each module is implemented as internal transformations on the XML data structure.

4.1 Formulate Requests

The first step in answering natural language questions in the knowledge mining component is to translate them into Aranea queries, or requests. These requests specify the textual context in which answers are likely to be found, and are analogous to queries posed to information retrieval systems. However, because Aranea relies on Web search engines to fulfill these requests, fine-grained control over the result set is impossible. Aranea instead relies on *quantity* to make up for lack of *quality*.

Two types of queries are generated by this module: *exact* (or *reformulation*) queries and *inexact* (or *back-off*) queries. Queries of both types are concurrently generated, but usually given different scores.

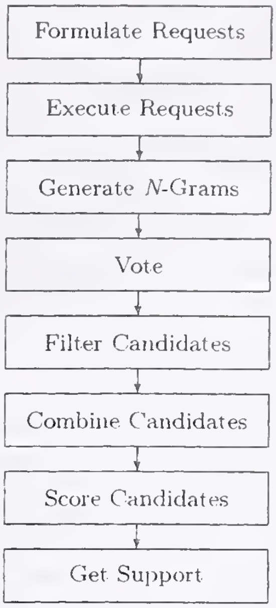


Figure 3: Data flow in the knowledge mining component of Aranea

An inexact query indicates that an answer is likely to be found within the vicinity of a set of keywords. They are composed by treating the natural language question as a bag of words.

An exact query specifies the location of a potential answer in more detail, e.g., the answer to "When did the Mesozoic period end?" is likely to appear within ten words and fifty bytes to the right of the exact phrase "the Mesozoic period ended". Exact queries in Aranea are generated by approximately a dozen pattern matching rules based query terms and their part-of-speech tags; morpho-lexical pattern matches trigger the creation of reformulated exact queries. As an example, the previous query was generated by the rule "*wh-word* did ... *verb* → ... *verb*+ed". An internal lexicon ensures that the generated verb remains properly inflected.

As a complete example, the requests generated in response to the question "When did the Mesozoic period end?" are shown in Figure 4. Aranea generates two inexact and one exact requests; each request is assigned a basic score, which establishes the relative importance of the queries.

4.2 Execute Requests

The request execution module is responsible for retrieving textual "snippets" that honor the constraints set forth in each request. Currently, the Google search engine is used to mine text from the

[2] Unfortunately, this technique equates the *most popular* answer with the *correct* answer, which occasionally results in very comical responses.

[3] as of early 2003

Query: When did the Mesozoic period end
Type: inexact
Score: 1
Number of snippets to mine: 100

Query: the Mesozoic period ended
Type: inexact
Score: 1
Number of snippets to mine: 100

Query: the Mesozoic period ended ?x
Type: exact
Score: 2
Number of snippets to mine: 100
Maximum length for `?x`: 50
Maximum word count for `?x`: 5

Figure 4: Typical requests generated by Aranea.

Web. In the case of inexact requests, the entire summary provided by Google is extracted for further processing. For exact queries, the request execution module performs additional pattern matching to ensure that the correct positional constraints are satisfied.

4.3 Generate *N*-Grams

This module exhaustively generates all possible unigrams, bigrams, trigrams, and tetragrams from the text fragments generated by the request execution module. These *n*-grams, which are given initial scores equal to the weight of the request from which they derive, serve as the raw candidate answers.

4.4 Vote

The voting module collates the *n*-grams generated by the previous module. The new score of each answer candidate is equal to the sum of the scores of all occurrences of that particular *n*-gram. This module has the effect of promoting text fragments that occur frequently (in the context of query terms), and are hence more likely to answer the user question.

4.5 Filter Candidates

In this stage of the processing, a coarse-grained filter in applied to answer candidates:

- Candidates that begin or end with stopwords are discarded.
- Candidates that contain words found in the user question are discarded. The only exception to this rule is question focus words, e.g., a question beginning with "how many meters..." can be answered by an expression containing the word *meters*.

In addition, this stage encodes a few heuristics that can potentially decrease the number of answer candidates. For example, the answer to "how far", "how fast", "how tall", etc., questions must contain a numeric component (either numeric digits or numerals); thus, we can safely discard all answer candidates that do not fit these criteria. We have also noticed that "who" and "where" questions usually cannot be answered with expressions that contain tokens consisting of numeric digits; Aranea can similarly reduce the number of answer candidates based on this criterion. The general principle embodied in this module is to filter with high confidence, erring on the side of being too lenient. False positives can always be sorted out by later modules, but the system will not be able to recover from false negatives.

4.6 Combine Candidates

In this module, shorter answers are used as evidence to boost the score of longer answers. If a portion of a candidate answer appears itself as a candidate answer, then the score of the shorter answer is added to the score of the longer answer. For example, if "de Soto" appears on the list of candidate answers along with "Hernando de Soto", the score of the shorter candidate would be added to the score of the longer one. This module counteracts the tendency of the *n*-gram generation and voting modules to favor shorter answers.

4.7 Score Candidates

The score of each answer candidate is multiplied by the following factor:

$$\frac{1}{|A|}\sum_{w\in A}\log(\frac{N}{w_c})$$

A is a set of keywords in the candidate answer; N is the total number of words in the AQUAINT corpus; w_c is the number of occurrences of word w in the AQUAINT corpus. Each answer candidate is scaled by the average of an *idf*-like value of its component keywords. This scoring balances the effect of individual keywords having different (unconditioned) priors. Since the exact distribution of unigrams on the Web can not be easily obtained in a reliable manner, Aranea uses statistics from the AQUAINT corpus as a surrogate.

4.8 Get Support

This module performs a final sanity check on the candidate answers. It verifies that final candidate answers actually appear in the original text snippets mined from the Web. Occasionally, the various modules within the knowledge mining component of the system will assemble a nonsensical answer; this module ensures that such answers are discarded.

5 Answer Boosting and Cleanup

Results from both the knowledge annotation and knowledge mining components of Aranea are subjected to a series of heuristic checks. These heuristics may employ external knowledge resources to verify the candidate answers.

The answer boosting module of Aranea contains heuristics specifically dedicated to verifying geographic locations. We have gathered large lists of known geographic entities, e.g., world cities, US cities, etc.; these lists allow us to "boost" the score of certain answer candidates in response to "what city", "what state", "what country", "what province", etc. questions.

Questions requiring dates as answers similarly receive special treatment. Named entity detectors allow us to promote dates over other noun phrases. Knowledge of dates also helps Aranea extract the exact answers. For example, a candidate answer to a "what year" question often contains extra information such as the month and day; Aranea removes such extraneous information.

Beyond a few simple heuristics, Aranea also performs part-of-speech tagging on the answer candidates to ensure that they are full constituents (NP or VP). Extra leading or trailing words are trimmed.

6 Answer Projection

The final step in the preparation of an answer derived either from knowledge annotation or knowledge mining is answer projection, during which each Web-extracted answer is paired with a document from the AQUAINT corpus to form the basic [*answer*, *docid*] response unit. Answer projection was accomplished in a two step process: first, a set candidate documents was gathered; then, a modified passage retrieval algorithm scanned the documents to pick the best document.

We experimented with three different methods of retrieving a candidate set of documents on which to project our Web-derived answers:

- **NIST documents.** The top fifty documents supplied by NIST served as the baseline set of candidate documents for answer extraction.
- **MultiText passages.** We have implemented the passage retrieval algorithm described by Clarke *et al.* (2000). A set of passages generated by this algorithm serves as the candidate documents for answer projection.
- *PC3* **MultiText passages.** We have augmented the MultiText passage retrieval algorithm by a backoff procedure we call *pc3*. Our algorithm applies a series of controlled query expansion loops, which successively broadens the query terms (e.g., by including different inflections and synonyms of the keywords) until an adequate set of candidate passages have been found.

After a set of candidate documents has been gathered, the answer projection module applies a modified window-based passage retrieval algorithm to score the documents. Each 140-byte window is given a score equal to the number of times keywords from both the question and candidate answer appears, with the restriction that at least one keyword from the question must appear in the particular passage. The score of a document is simply the score of the highest scoring passage. The highest scoring document is paired with the Web-derived candidate answer as the final response unit.

7 Confidence Ordering

This year's TREC evaluation required participants to sort answers according to confidence, motivated by the importance of a system knowing when it is likely to be right or wrong. Although this was certainly an interesting aspect of the question answering task, due to time constraints, we were unfortunately not able to devote much attention to it.

For the deployed TREC system, we employed a crude algorithm:

- All *when* questions were placed before all *who* and *where* questions, which were ordered before all *what* questions. All other questions were placed after. We discovered through ad-hoc experimentation that Aranea generally performed better on certain types of questions; the confidence ordering reflected our experiences.
- Within each type of question, answers derived from knowledge annotation were always placed before answers derived from knowledge mining.
- Answers were sorted by the document score produced by the answer projection algorithm
- Any further ties were broken by scores generated by the knowledge mining component.

8 Results

The official TREC results are shown in Table 1. The only difference between our three runs was the method used to generate the initial set of candidate documents for answer projection:

- `aranea02a` used only the top fifty NIST-supplied documents.
- `aranea02pbq` used the top fifty NIST-supplied documents and passages derived from the MultiText algorithm.
- `aranea02pc3` used the top fifty NIST-supplied documents and passages derived from the *pc3* variant of the MultiText algorithm.

In addition, we analyzed Aranea's performance without taking into account answer projection. We felt that this particular instance of answer projection is an artifact of the TREC evaluation, and not

		aranea2002a		aranea2002pbq		aranea2002pc3		aranea2002	
		NIST Docs		MultiText		*pc3* MultiText		no projection	
Knowledge Annotation	correct	22	4.4%	23	4.6%	22	4.4%	30	6.0%
	inexact	2	0.4%	3	0.6%	2	0.4%	2	0.4%
	unsupported	8	1.6%	6	1.2%	8	1.6%	-	-
	wrong	10	2.0%	10	2.0%	10	2.0%	10	2.0%
	total	42	8.4%	42	8.4%	42	8.4%	42	8.4%
Knowledge Mining	correct	130	26.0%	131	26.2%	129	25.8%	153	30.6%
	inexact	34	6.8%	33	6.6%	34	6.8%	43	8.6%
	unsupported	32	6.4%	32	6.4%	32	6.4%	-	-
	wrong	262	52.4%	262	52.4%	262	52.4%	262	52.4%
	total	458	91.6%	458	91.6%	458	91.6%	458	91.6%
Total	correct	152	30.4%	154	30.8%	151	30.2%	183	36.6%
	inexact	36	7.2%	36	7.2%	36	7.2%	45	9.0%
	unsupported	40	8.0%	38	7.6%	40	8.0%	-	-
	wrong	272	54.4%	272	54.4%	273	54.6%	272	54.4%
	total	500	100%	500	100%	500	100%	500	100%
	CWS score	**0.433**		**0.427**		**0.421**		**0.529**	

Table 1: TREC Results

		aranea2002a	aranea2002pbq	aranea2002pc3	aranea2002
		NIST Docs	MultiText	*pc3* MultiText	no projection
Knowledge Annotation	correct	52.4%	54.8%	52.4%	71.4%
	inexact	4.8%	7.1%	4.8%	4.8%
	unsupported	19.0%	14.3%	19.0%	-
	wrong	23.8%	23.8%	23.8%	23.8%
Knowledge Mining	correct	28.4%	28.6%	28.2%	33.4%
	inexact	7.4%	7.2%	7.4%	9.4%
	unsupported	7.0%	7.0%	7.0%	-
	wrong	57.2%	57.2%	57.4%	57.2%

Table 2: Performance of individual components.

inherent in the question answering task itself. We rescored the unsupported judgments of aranea02a either as inexact or correct, careful to adhere to the same standards of judgment as the other runs. This result is shown in the last column of Table 1.

In the formal TREC runs, our system answered approximately thirty percent of the questions correctly. Disregarding answer projection, Aranea provided exact, correct answers for nearly thirty-seven percent of the questions. Out of five hundred questions, 42 (8.4%) answers were contributed by Aranea's knowledge annotation component; the knowledge mining component accounted for the rest, or 458 (91.6%) questions.

Approximately 15% of answers judged as correct were derived from knowledge annotation techniques. We believe that this performance is remarkable, considering that our system contained only twenty-eight data access schemata over seven sources, representing no more than a few person-days worth of knowledge engineering effort. Our experiences with Start and Omnibase have helped us streamline the knowledge engineering process, allowing us to rapidly structure knowledge sources to answer English questions. These results also verify that analysis of the typical distribution of user questions can help guide the knowledge engineering effort. Our database access schemata were geared towards answering the most frequently occurring questions from the previous TREC evaluations; many of the same question types also appeared in this year's evaluation.

Overall, we noticed that answer projection was the obvious weak link in Aranea. For approximately twenty percent of our Web-derived answer, our system was unable to find an adequate supporting document, which resulted in a drastic reduction of our overall TREC score. Our passage retrieval algorithm was not sophisticated enough to ignore documents that contained keywords from the question and answer, but in fact did not answer the question. More future research is required to obtain better answer projection performance.

Individual analysis of each Aranea component is shown in Table 2. In general, the database component achieves much higher accuracy than the knowledge mining component, due to the knowledge engineering effort involved in creating database access schemata. However, projecting answers derived from database access appears more difficult than answers derived from knowledge mining. Once again, we believe that Aranea demonstrates the validity and effectiveness of knowledge engineering in the question answering process. Knowing when to apply manual effort and selectively using human labor can translate into a big payoff in terms of performance enhancement.

9 Contributions

The Aranea system presents two different paradigms for approaching the question answering problem. In the knowledge annotation approach, natural language questions can be translated into database queries, which then extract answers from the Web. In the knowledge mining approach, data redundancy on the Web can be leveraged to overcome many difficult problems in natural language processing.

Aranea smoothly integrates both the knowledge annotation and knowledge mining approach into a uniform framework. With knowledge about the types of questions that users ask, we were able to utilize each paradigm effectively.

Another insight we gained in developing Aranea is to let the analysis of user questions guide our knowledge engineering effort. By correctly anticipating the types of questions users typically ask, we were able to construct effective database access schemata with reasonable amounts of manual labor.

We believe that Aranea provides a well-engineered platform for experimenting with various Web-based question answering techniques. In the future, we will continue to refine existing technology and develop new methods for answering natural language questions.

10 Acknowledgements

This research is funded by DARPA under contract number F30602-00-1-0545 and administered by the Air Force Research Laboratory. Additional funding is provided by the Oxygen Project.

References

Eric Breck, Marc Light, Gideon S. Mann, Ellen Riloff, Brianne Brown, Pranav Anand, Mats Rooth, and Michael Thelen. 2001. Looking under the hood: Tools for diagnosing your question answering engine. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL'01) Workshop on Open-Domain Question Answering.*

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. 2001. Data-intensive question answering. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001).*

Eric Brill, Susan Dumais, and Michele Banko. 2002. An analysis of the AskMSR question-answering system. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002).*

Sabine Buchholz. 2001. Using grammatical relations, answer frequencies and the World Wide Web for question answering. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001).*

Charles Clarke, Gordon Cormack, Derek Kisman, and Thomas Lynam. 2000. Question answering by passage selection (multitext experiments for TREC-9). In *Proceedings of the Ninth Text REtrieval Conference (TREC-9).*

Charles Clarke, Gordon Cormack, and Thomas Lynam. 2001. Exploiting redundancy in question answering. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-2001).*

Eduard Hovy, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2002. Using knowledge to facilitate factoid answer pinpointing. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING-2002).*

Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin, Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran. 2002a. Omnibase: Uniform access to heterogeneous data for question answering. In *Proceedings of the 7th International Workshop on Applications of Natural Language to Information Systems (NLDB 2002).*

Boris Katz, Jimmy Lin, and Sue Felshin. 2002b. The START multimedia information system: Current technology and future directions. In *Proceedings of the International Workshop on Multimedia Information Systems (MIS 2002).*

Boris Katz. 1988. Using English for indexing and retrieving. In *Proceedings of the 1st RIAO Conference on User-Oriented Content-Based Text and Image Handling (RIAO '88).*

Boris Katz. 1997. Annotating the World Wide Web using natural language. In *Proceedings of the 5th*

RIAO Conference on Computer Assisted Information Searching on the Internet (RIAO '97).

Cody Kwok, Oren Etzioni, and Daniel S. Weld. 2001. Scaling question answering to the Web. In *Proceedings of the Tenth International World Wide Web Conference (WWW2001).*

Jimmy Lin and Boris Katz. 2003. Question answering techniques for the World Wide Web. In *EACL-2003 Tutorial.*

Jimmy Lin, Dennis Quan, Vineet Sinha, Karun Bakshi, David Huynh, Boris Katz, and David R. Karger. 2003. The role of context in question answering systems. In *Proceedings of the 2003 Conference on Human Factors in Computing Systems (CHI 2003).*

Jimmy J. Lin. 2002. The Web as a resource for question answering: Perspectives and challenges. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002).*

John B. Lowe. 2000. What's in store for question answering? (invited talk). In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000).*

Martin M. Soubbotin and Sergei M. Soubbotin. 2001. Patterns of potential answer expressions as clues to the right answers. In *Proceedings of the Tenth Text REtrieval Conference (TREC 2001).*

MITRE's Qanda at TREC-11*

John D. Burger, Lisa Ferro, Warren Greiff,
John Henderson, Marc Light†, Scott Mardis, Alex Morgan

The MITRE Corporation
†The University of Iowa

{john, lferro, greiff, jhndrsn, mardis, amorgan}@mitre.org
marc-light@uiowa.edu

Introduction

Qanda is MITRE's TREC-style question answering system. Since last year's evaluation, principal improvements to the system have been aimed at making it faster and more robust. We discuss the current architecture of the system in Section 1. Some work has gone into better answer formation and ranking, which we discuss in Section 2. After this year's evaluation, we have done a number of ROVER-style system combination experiments using the judged answer strings made available by NIST. We report on some success with this in Section 3. We have also performed a detailed categorization of previous TREC results according to answer type and grammatical category, as well as an analysis of Qanda's own question analysis component—see Section 4 for these analyses.

1. TREC-11 System Description

Catalyst

Last year, Qanda was re-engineered to use a new architecture for human language technology called *Catalyst*, (Burger & Mardis 2002). Developed at MITRE for the DARPA TIDES program, the Catalyst architecture is specifically designed for fast processing and for combining the strengths of Information Retrieval (IR) and Natural Language Processing (NLP) into a single framework. Catalyst uses a dataflow architecture in which standoff annotations are passed from one component to another, the components being connected in arbitrary topologies (currently restricted to acyclic ones). The use of standoff annotations permits components to be optimized for just those pieces of information they require for their processing.

Major system components

Qanda has a by now familiar architecture—questions are analyzed for expected answer types, documents are retrieved using an IR system and are then processed by various taggers to find entities of the expected type in contexts that match the question. Below we describe each of the major components in turn.

Question analysis: This component is run after the question has been subjected to POS and named entity tagging. It uses a simple grammar, currently hand-written, to identify important components of the question—see Section 4 below for more detail.

IR wrappers: Catalyst components have been written for several IR engines, taking the results of the question analysis and formulating an IR query. For TREC-11, we used the Java-based Lucene engine (Apache 2002). Lucene's query language has a phrase operator, and also allows query components to be given explicit weights. Qanda uses both of these capabilities in constructing queries from the information extracted from the question. For TREC-11, the top 25 documents were retrieved.

Passage processing: Retrieved documents are tokenized, and sentence boundaries are detected. Because some downstream components run more slowly than the rest of the system, Qanda scores each sentence by summing the log-IDF (inverse document frequency) of each word that overlaps with the question. Only those sentences with a sufficient score are passed on to the rest of the system.

Named entity tagging: Qanda uses Phrag (Burger et al. 2002), an HMM-based tagger, to identify named persons, locations and organizations, as well as temporal expressions. Phrag is also used as a POS tagger for question analysis.

Numeric tagging: A simple pattern-based tagger uses an extensive list of unit phrases to identify measures, as well as currency, percentages and other numeric phrases.

*This work was supported in part by the Advanced Research and Development Activity (ARDA) under contract number contract F19628-99-C-0001.

Other taggers: We have a simple facility for constructing taggers from fixed word- and phrase-lists. These were used to re-tag many named locations more specifically as cities, states/ provinces, and countries. Qanda also identifies various other (nearly) closed classes such as precious metals, birthstones, various animal categories, etc.

Answer formation and ranking: Candidates are identified and merged, a number of features are collected, and a score is computed—see Section 2.

Qanda's answer formation component attempts to find the best answer phrase as well as the best supporting context for that answer—the former may not be a substring of the latter due to candidate merging. For our TREC-11 submission, the answer phrase was used as the actual (scored) answer string, while the context was included as the secondary (unscored) justification.

2. Answer Ranking

Qanda only examines sentences that match the question sufficiently using the IDF-weighted overlap described above. It collects candidate answers by gathering phrasal annotations from all of the semantic taggers, and identifies the following features:

Context IDF Overlap: Described above.

Context Bigram Overlap: Raw count of word bigrams in common with the question.

IR Ranking of the source document by the IR system.

Type Same: Boolean, true if the candidate and expected answer types are identical.

Type Similar: Partial credit if the candidate's type is "related" to the expected answer type, e.g., *COUNTRY* and generic *LOCATION*.[1]

Candidate Overlap: Raw count of non-stop words in common between the candidate itself and the question, to bias against entities from the question being chosen as answers.

Minimal Overlap Distance: Number of characters between the candidate and the closest non-stop question word in the context.[2]

Numeric Date: 1 if the expected answer type is temporal and the candidate contains a numeric token, 0 otherwise, to bias against unresolved relative dates such as *yesterday*.

Candidates with similar textual realizations are merged, with the combined candidate retaining the highest value for each feature. Accordingly, an additional feature is maintained:

Merge Count

After all of the (merged) candidates have been acquired, the raw feature values described above are normalized with respect to the maximum across all candidates, resulting in values between 0 and 1.[3] Features normalized in this way are more commensurate across questions, especially word overlap and related features (Light et al. 2001).

Each feature has a fixed weight, and a simple additive model is used to give each candidate an overall score. Our official TREC submission used (minimally) hand-tuned weights.

Log-linear models for answer scoring and confidence estimation

We are currently experimenting with acquiring the weights for the answer scoring component using logistic regression on past TREC datasets, resulting in a log-linear model, as has been used by Ittycheriah et al. (2001) and others. One issue is that because the model estimates a conditional probability (namely correctness given features of the question and candidate), the resulting scores are not necessarily commensurate across questions, and so the answers cannot be easily ranked for confidence, as required in TREC this year. Our current approach is to re-score the top candidate for each question using a second log-linear model. This uses features of the question, such as expected answer type, that do not affect the first model's candidate ranking, as well as features derived from applying the first model, such as the top candidate's original score, the total score mass given to the top *N* candidates, etc. These last features are similar to those used by Czuba et al. (2002).

[1] Currently this is done using a simple hand-built table, but with sufficient training data, we expect to use the log-linear model described below to acquire weights for most sensible pairs of types.

[2] Words would arguably be a more intuitive unit for this feature.

[3] The normalized values are computed so that the intuitively "best" feature value is 1, the worst 0—this is primarily for the developers' convenience, but also so weights are all positive, and more easily reasoned about.

3. System-Combination Experiments—Exploiting Diversity

Progress in question answering technology can be measured as individual systems improve in accuracy, but it is not the only way to witness technological progress. A question one can ask is how well we can perform automatic question answering *as a community*. If we were asked to enter an Earth English system in an intergalactic TREC, how well would we do? One easy answer is that we would perform as well as the best QA system. A second answer is that perhaps we could do even better by combining systems—this might be expected to work if different systems were independent in their errors. The follow-up question is how would we build such a system?

Lower bounds on the highest possible performance current technology can achieve on a given dataset have practical value, as well. They allow us to better estimate how well systems are doing with respect to the underlying difficulty of the dataset, and continually provide performance targets that are known to be achievable. Without such lower bounds on optimal performance, one cannot determine if technological progress in a domain has simply stalled.

NIST's ROVER system for combining speech recognizer output gives ASR researchers an updated goal to shoot for after every evaluation, as well as an implicit measure of the extent to which systems are making the same errors (Fiscus 1997). The work herein initiates a similar set of experiments for question answering technology.

Methods

The task we are faced with is straightforward. Given a collection of answers to a question, choose the one most likely to be correct. For our purposes, each answer consists of the answer string and an identifier for an associated document. Our data was limited in that it did not indicate which answers were provided by which system—see the discussion below. Note that we use no knowledge of the question or of the document collection. Our assumption is that the authors of the individual systems have milked the information in their inputs to the best of their capabilities. Our goal is to combine their outputs, not to re-investigate the original problem.

In this year's main QA evaluation there were 67 different systems or variants thereof involved. Thus, our corpus consists of 67x500 answers. To guard against any implicit bias due to repeated experimentation on the small dataset available, we randomly selected a 100-question subset for development of our techniques—the remaining 400 questions were kept as a test set, evaluated only once, when development was complete. While we may have wished to pursue parametric techniques, we felt that this training set was too small to explore any but the simplest (non-parametric) techniques. An exception is the experiments described below involving priors over the document sources, which still only involved four parameters.

Voting is an easily understood technique for selecting an answer from among the 67 suggestions. Unfortunately, voting techniques do not provide a mechanism for utilizing full knowledge of partial matches between proposed answers. While his original goal was the selection of representative DNA sequences, Gusfield (1993) introduced a general method for selecting a candidate sequence that is close to an ideal centroid of a set of sequences. His technique works for all distance measures that support a triangle inequality, and offers a bound that the sum of pairwise distances (SOP) from proposed answers to the chosen answer will be no more than twice the SOP to the actual centroid (even though the centroid may not be in the set). This basic technique has been used successfully for combining parsers (Henderson 1999). Appealingly, the centroid method reduces to simple voting when an "exact match" distance is used (the complement of the Kronecker delta).

One advantage of both simple voting and the centroid method is that they give values (distances) that are comparable between questions. An answer that receives 20 votes is more reliable than an answer that receives 10 votes, and likewise for generalized SOP values. This gives a principled method for ranking results by confidence and measuring average precision, as required for this year's TREC evaluations.

In selecting appropriate distance measures between answers, both words and characters were explored as atomic units of similarity. Two well-known non-parametric distances are available in the literature: Levenshtein edit distance on strings and Tanimoto distance on sets (Duda et al. 2001). We experimented with each of these, and also generalized the Tanimoto distance to handle multisets by defining a function to map multisets to simple sets: Given a multiset containing instances of a repeated element x we can create a simple set by subscripting, e.g., $<x,x,y,z> \Rightarrow \{x_1,x_2,y,z\}$. We can then use the standard Tanimoto

	Dev Set (100 Qs)				**Test Set (400 Qs)**	
	Strict		**Loose**		**Strict**	
	P	**avgP**	**P**	**avgP**	**P**	**avgP**
exact string match	50	70	54	74	42	65
word set	54	75	58	78	46	68
word bag	54	75	58	78	46	68
character set	51	65	57	67	46	62
character bag	60	81	64	85	50	**74**
word bag w/ doc priors	**66**	83	74	88	51	72
character bag w/ doc priors	64	81	69	86	50	72
5-character bag w/ doc priors, weighted numeric strings	**66**	**85**	**76**	**90**	**53**	73

Figure 1: Answer selection results (percentages, best results in bold)

distance on the resulting simple sets.[4]

Overall, systems seemed to be conservative and answered with the NIL document (no answer) at a rather high rate (17% of all answer strings this year). To compensate for this, a "source prior" was collected from the 100-question training set. These four numbers recorded the accuracy expected when systems generated answers from the four document sources (AP, NYT, XIE, and NIL). Those numbers were then used to scale the distance measures for the corresponding answer strings. Other than these priors, no other features of the document ID string were used.

Results

Several measurements were made to ascertain the quality of the various selection techniques, as seen in Figure 1. Precision, **P**, indicates the accuracy of the technique, the percentage of the answers that were judged to be correct. **avgP** is the main measure used by NIST this year—the average precision of all prefixes of the sequence of answers placed in order of high to low confidence. **Strict** corresponds to the correctness criterion used by NIST—the answer must be exact and justified by the referenced document (assessor judgment 1). The **Loose** figures discard these two criteria (assessor judgment ≥ 1). The **Loose P** measure was the one that was optimized during development.

In Figure 1 we see both development and test set results for answer selection experiments involving a sample of the distance measures with which we experimented. All of the design and selection of the distance measures was done using hill-climbing on the development set, and only after this exploration was complete was the performance on the test set measured. Two general observations can be made about these results (and others not shown): taking into account a prior based on the document source (including NIL) is useful, as is working with feature bags from the answers rather than sets. The best-performing selection system used all character strings of length 5 and less as features, combined with the multiset Tanimoto distance measure described above, and scaled with document source priors. Furthermore, a numeric string mismatch was weighted to be twice as costly as mismatching a non-numeric string. Question 1674 provides an example that contrasts this best selector with a simple voting scheme (exact string match):

> *What day did Neil Armstrong land on the moon?*
> *1969* (simple voting—incorrect)
> *July 20, 1969* (best measure above—correct)

While a plurality of systems answered with *1969*, many others answered with variants of the correct answer that differed in punctuation, as well as *on July 20, 1969*; *July 18, 1969*; *July 14, 1999*; even simply *20*. All of these, including the incorrect *1969*s, contributed to the correct answer being selected.

The disparity between the dynamic range of these systems on the development dataset and the test dataset suggests that the dev set sample size of 100 (6700 proposed answers and NILs) is too small to draw conclusions on the relative quality of selection techniques. Still, consistencies in rank orderings of selection techniques between the two datasets strongly suggest that these methods of system combination are effective.

It is important to note that in these experiments we did not have access to several useful evidence sources. First, this year's submissions included system

[4] Recall $D_T(S_1, S_2) = 1 - |S_1 \cap S_2| / |S_1 \cup S_2|$.

estimates on answer confidence, if only implicitly. The selection mechanism could take advantage of this by weighting each submitted answer string appropriately. Second, past TRECs show that some systems are reliably more accurate than others, and if each answer string were labeled with a system ID, even if anonymized, we could use system-level features in the selector, such as a simple prior. Given sufficient training, we might even take question features into account, learning that certain systems are better at certain types of questions. We would like to pursue the use of these and other evidence sources in the future.

4. Analysis of Questions and Answers

In this section, we report on a number of analyses we have performed, both on Qanda and on all-system results from previous TRECs. We describe the features Qanda extracts from questions, and evaluate its performance on one of these. We also describe a detailed categorization of the TREC-9 answer corpus with respect to semantic and syntactic types.

Question analysis in Qanda

Phrag, our HMM-based tagger, first annotates questions using separate models for part-of-speech and named entities. Qanda also runs a simple lookup-based tagger that maps head words to answer types in Qanda's ontology using a set of approximately 6000 words and phrases, some extracted semi-automatically from WordNet, some identified by hand. Based on these annotations, Qanda's main question analysis component uses a parser with a simple hand-optimized grammar to identify the following aspects of each question:

> Answer type: a type in Qanda's (rather simple) ontology, e.g., *PERSON* or *COUNTRY*.
>
> Answer restriction: an open-domain phrase from the question that describes the entity being sought, e.g., *first woman in space*.
>
> Salient entity: What the question is "about". Typically a named entity, this corresponds roughly to the classical notion of *topic* discussed below, e.g., *Matterhorn* in *What is the height of the Matterhorn?*
>
> Geographical restriction: Any phrase that seems to restrict the question's geophysical domain, e.g., *in America*.
>
> Temporal restriction: Any phrase that similarly restricts the relevant time period, e.g., *in the nineteenth century*.

As part of our post-TREC analysis, we have begun to examine how well Qanda performs on these various aspects. One way of evaluating this is to create an annotated test set of questions, tagged with the "correct" result, and score Qanda against this. For example, we might annotate *When did the art of quilting begin?*, with the answer type *LOCATION*—if Qanda's prediction matches this, its question analysis was correct in this instance. However, there is another approach to this evaluation. As described in the next section, we have annotated the TREC-9 answer key with semantic types, and so one might ask how often the system predicts one of the actual answer types, according to the answer key. For our example question, *medieval Europe*—a *LOCATION* answer—was judged to be correct by the TREC assessors. Had this been the only correct answer found, Qanda's prediction would be counted wrong, under the analysis we describe here.

In Figure 2 we present this analysis in terms of the question phrase used, and as a percentage of all questions in the set. This helps us to see which question types might have the biggest impact on our performance. For example, Qanda does rather well at predicting an answer type for *how many* questions, but these only constitute 5.44% of the questions in the set. On the other hand, of the 29.71% of the set that were unadorned *what* questions, Qanda's question

	Correct	Incorr.	Total
at what X	0.23	0.00	0.23
for what X	0.00	0.23	0.23
in what X	0.00	0.23	0.23
what in-situ	0.00	0.45	0.45
what kind	0.00	0.68	0.68
what type	0.00	0.68	0.68
what X	5.90	5.90	11.79
what	3.17	26.30	29.71
how hot	0.00	0.45	0.45
how large	0.00	0.23	0.23
how long	0.00	0.68	0.68
how many	5.22	0.23	5.44
how much	1.13	0.00	1.13
how tall	0.00	0.45	0.45
how wide	0.23	0.00	0.23
name	0.23	0.00	0.23
tell	0.00	0.23	0.23
when	9.07	0.00	9.07
where	12.70	0.91	13.61
who	20.41	1.59	22.00
why	0.00	0.23	0.23
Grand Total	58.50	41.27	100.00

Figure 2: Answer type correctness (percentage of all questions)

component performed very poorly. We hope to perform similar evaluations for the other question aspects listed above.

Manual answer analysis of the TREC-9 question corpus

Here we report on an analysis of the answers returned by all systems participating in TREC-9. Our study was done as part of a larger investigation, consisting of two levels: First, to identify Topic and Focus constituents for each question, and second, to characterize the Topic and Focus constituents by referent, and in the case of certain expressions, by grammatical type.

Before we explain what each of these levels of analysis entailed, we will first establish what we mean by Topic and Focus, as the terms and concepts are often used interchangeably in the Q&A literature. We use the terms Topic and Focus as they are defined in classic formal linguistics, dating back to the mid 19th century (see Hajicova 1984, for an early historical overview) and continuing on to recent times in linguistic schools such as Functional Grammar (Dik et al. 1981) and generative grammar (Rochemont 1986). Variably termed theme/rheme, topic/comment, presupposition/focus, they are defined in discourse theory roughly as follows:

Topic: The constituent(s) of a sentence identifying given, presupposed, or "old" information at the time of the utterance.

Focus: The constituent(s) of a sentence identifying what is new to the discourse at the time of the utterance.[5]

In questions, the *wh*-word is the Focus, and the rest of the question is typically the Topic. The answer to a question is also Focused. Question/Answer pairs have long been used in traditional Topic/Focus research papers to unambiguously illustrate and identify Topic and Focus constituents. E.g., from Dik et al. (1981):

(1) question: (a) ***What** did John buy?*
answers: (b) *John bought **an umbrella**.*
(c) ***an umbrella***

Bold is used in (1) to identify the Focus constituents; normal weight text indicates Topic constituents. Ordinarily, utterances such as (1a) would occur in a context in which John's buying activity were already presupposed. Earlier models of discourse did not anticipate the context in which humans would be entering factual questions into computers "out of the blue." However, since TREC has yet to intentionally introduce questions with false presuppositions, in our analysis we assumed the presuppositions were true and considered them Topic constituents.

Returning to the discussion of the analysis of the TREC question set, we identified the Topic and Focus constituents of each question, for example:

(2) <FOCUS>*Who*</FOCUS> <TOPIC>*invented the paper clip*</TOPIC>*?*

In addition, we used a REF attribute to classify the entity or activity REFerenced by the constituent, where the value for REF comes from an entity/activity ontology, shown in Figure 3 below. For certain expressions, we also used an EXP attribute to identify whether the EXPression is a name, descriptor, or directional phrase. Except for cases requiring a "direction" value (see example 5), EXP is typically only used for classic "Named Entities" such as persons, locations and organizations. Artifacts will also sometimes have an EXP attribute. Here is the previous example with these attributes marked:

(3) <FOCUS REF="person" EXP="name">
Who</FOCUS> <TOPIC REF="levin_26_4">
invented the paper clip</TOPIC>*?*

The markup in example 3 identifies the answer to this question as a named person and identifies the Topic of the question as a creation activity (levin_26_4 is the class of *create* verbs.) The annotation of the Topic constituents in the TREC-9 questions has not been finalized at this time, so in the remainder of this section we will discuss only the results of the Focus tagging.

In determining the value for REF and EXP in Focus constituents, we looked aι the actual answers as recorded in an answer key we developed previously. This answer key[6] was compiled by manually examining all the answers returned by all of the TREC-9 systems. From those judged correct by the TREC assessors, we extracted short answer phrases. To perform the Focus analysis, we annotated the answer key itself, rather than the *wh*-word as shown in example 3, because there are often multiple correct answers to a given question.[7] We tagged each possible

[5] There are actually two types of Focus: Completive and Contrastive. Here we refer only to Completive Focus.

[6] See http://trec.nist.gov/data/qa/add_qaresources.html

[7] Multiple answers are due to two factors: different phrasings of the same correct answer and completely different correct answers. We did not distinguish between these two factors in our analysis of the answers.

- **Entity**
    - organism
        - person (includes deities)
        - animal (non-human)
        - plant
    - body_part
    - plant_part
    - organization
    - other_agent
    - celestial (e.g., Earth, Sun, Horsehead Nebula)
    - geological (e.g., mountain, river, continent, oceans)
    - gsp (Geo-Social political entity)
        - country
        - city (villages, towns)
        - province (counties, states)
    - recreational (e.g., parks, preserves, monuments)
    - other_location
    - facility
    - artifact
        - titled_work
            - book
            - movie
            - music
        - vehicle
        - award
        - instrument (musical)
    - substance
        - liquid
        - solid
        - gas
    - temporal
        - date
        - time

- **Abstract**
    - language
    - thought
    - utterance
        - translation
        - statement
        - description
        - question
    - technique
    - quantity
    - age
    - measure
        - mass
        - volume
        - area
        - length (height, etc.)
        - frequency (any type of rate)
        - temperature
        - weight
        - energy
        - illumination
        - duration
        - monetary
    - signal
        - appearance
            - color
            - shape
        - sound
        - sensation
        - flavor
        - scent

- **Disease**
- **Phenomenon** (e.g., physical phenomenon)
- **Manner** (e.g., slowly, well)
- **Mode** (by plane, by camel)
- **Event**
- **Activity**
    - Levin (1993) verb classes where possible, else FrameNet classes
- **Emotion** (feelings)
- **Stative** (being, having, spatial relations)
    - physiological (e.g., bodily symptoms such as fever and depression)
- **Nationality**
- **Weather** (e.g., rain, cloud, fog)

Figure 3: Entity and activity ontology for question analysis

answer as a Focus constituent, and applied the correct REF and EXP attributes. For example:

(4) *What is Francis Scott Key best known for?*
<FOCUS REF="levin_26_7">*penned the national anthem*</FOCUS>;
<FOCUS REF="music" EXP="descriptor">*the national anthem*</FOCUS>;
<FOCUS REF="music" EXP="name">*Star-Spangled Banner*</FOCUS>

(5) *Where did Woodstock take place?*
<FOCUS REF="city" EXP="name">*Bethel* </FOCUS>;
<FOCUS REF="city" EXP="direction">
50 miles from Woodstock</FOCUS>

Metonyms, dangling modifiers, and similar expressions can occur as answer phrases, creating the difficulty that the literal interpretation out of context, versus the intended referent within the given context, may be distinct. Thus, a third attribute, LITREF, identifies the entity or activity referred to by the phrase in isolation. REF is used for the intended referent in the context of the question. For example:

(6) *What is the most common cancer?*
<FOCUS REF="disease">*skin cancer*</FOCUS>;
<FOCUS REF="disease" LITREF="body_part">
skin</FOCUS>

(7) *Name an American made motorcycle.*
<FOCUS REF="vehicle" LITREF="organization">
Harley-Davidson</FOCUS>

Question corpus analyses

We took the annotation of the answer key and collapsed all identically tagged answers in order to identify the set of unique answer types associated with each question. We consider an answer type "unique" if it differs by all three attributes (REF, LITREF, and

	Question Phrase					
	who	what	when	where	how	name
Number of questions	102	231	40	60	48	15
Number of answer types	8	63	2	13	12	13
Average number of answer types per question	1.19	1.19	1.03	2.57	1.02	1.60
Percentage of questions with more than one answer type	16.67	14.72	2.50	68.33	2.08	33.33

Figure 4: Range of answer types by question type

EXP). Thus an answer of type *PERSON NAME* is considered distinct from answer of type *PERSON DESCRIPTOR*. We also categorized each question by its *wh*-phrase (question phrase) to provide a rudimentary form of question typing. Some of the patterns that emerged are presented and discussed below.

Figure 4 shows the range of answer entities/activities associated with the major question types in TREC-9. The *what* questions exhibit the highest number of different answer types (63), but only 14.72% of the individual *what* questions have more than one answer type. This is because, although *what* questions have as their foci a broad range of entities/activities, each individual question is typically concerned with only a particular entity or activity. For example *What is platinum?* has four different answer phrasings, but they all refer to an entity of type *SOLID*.

In contrast, the *where* questions utilize only 13 answer types, but 68.33% of the *where* questions have more than one answer type. This is largely explained by the range of granularity that is acceptable as an answer, where a geological area, country, state, or city can suffice, as well as what we called *direction* expressions like *110 miles northwest of New York City*. Thus the granularity of the entity ontology has an effect here; had we grouped all of these under a single *LOCATION* category, the number of answer types for *where* questions would be greatly reduced.

As stated above, we consider answer types unique if the form of the answer (EXP= name, descriptor, or direction) differs. However, for individual questions, it is not very common to have answer types that differ only by the expression form. *Where* questions, which can have three values for EXP, exhibit the most cases of this: of the 60 where questions, nine (15%) have duplicate REF values but unique EXP values. For example, *Where are diamonds mined?* is answered variously by country name, country descriptor, geological name, and geological direction. *Who* questions come in second, but fairly low; of the 102 *who* questions, eight (8%) have answer types that differ only by EXP (person name and person descriptor). Of the 231 *what* questions, only two have both organization name and organization descriptor, and only one has both person name and person descriptor.

Figure 5 shows the top ten answer types for *what* questions, and Figure 6 does the same for *where* questions. The **(no answer)** label in Figure 5 reflects

Answer Type	Percentage of *what* questions
organization	11.64
person	8.73
animal	6.18
artifact	5.45
date	4.36
disease	4.36
(no answer)	3.64
geological	3.64
quantity	3.64
city	3.27

Figure 5: Top ten *what*-question answer types

Answer Type	Percentage of *where* questions
city	19.48
country	18.83
geological	18.18
province	15.58
gsp	6.49
other_location	6.49
facility	5.19
recreational	4.55
organization	2.60
body_part	0.65

Figure 6: Top ten *where*-question answer types

	EXPression Type			
	name	**descriptor**	**direction**	**(no value)**
who	68.60	21.49	0.00	9.92
what	36.73	9.09	0.00	54.18
where	70.13	4.55	11.04	14.29
name	41.67	4.17	0.00	54.17

Figure 7: Expression types for selected question types (percentages)

questions for which there were no answers in the key, because no systems answered them correctly.

For *who* questions, 80.17% of the answers were of type *PERSON*, 9.09% were *ORGANIZATION*, and 4.96% had no answer. All but one of the *when* questions had a *DATE* answer type—*When did the art of quilting begin?* had *medieval Europe* (a *GSP*) as one possible answer. *Name* imperatives (see example 7 above) display a range of foci, but 42% fall into one of three categories: *VEHICLE* (16.67%), *ORGANIZATION* (12.5%), and *OTHER_LOCATION* (12.5%).

Finally, Figure 7 shows the common EXPression types for those questions that can be answered with names. Many answers lack an EXP value because they refer to entities that do not typically bear names. However, the high number of answers with no EXP values also reflects the preliminary nature of this annotation scheme, particularly for the *what* and *name* questions. While unambiguous names were marked consistently as such, we were conservative in the use of the DESCRIPTOR value until we could see what entities emerged from the data. In the future, we will be refining the guidelines to make better use of the DESCRIPTOR value, and perhaps expanding EXP to include other values like *ADJECTIVE* and *ADVERB*.

Other analyses of question corpora

There have been many previous efforts at classifying questions. We mention a few here for comparison purposes. Weischedel et al. (2002) reported on an analysis of the combined questions of TREC-8, 9 and 10. They found a prevalence of people, locations, countries/cities/states, and definitions. Their cumulative results for all three TRECs are not directly comparable to what we've reported here, due to differences in the ontologies used, and also because our analysis is based on an examination of the answers rather than the questions. Hovy et al. (2000) use an ontology similar to the one in Figure 3. But where our ontology is used to characterize the Topic and Focus constituents, theirs represents the user's intention in asking the question, so that the ontology includes categories like *Why-Famous*. Thus, similar-looking tactics can have very different underlying approaches; One future goal is to apply multiple approaches to the same corpus, for a richer understanding of questioning and answering phenomena.

5. Conclusion

As well as the requisite description of this year's system architecture, we have discussed some preliminary work on log-linear models for answer selection and confidence estimation. We would like to pursue this further, using more features and more sophisticated models. We also presented promising initial results on question answering system combination—we will be exploring this further, hopefully making use of system-specific priors as well as confidence information in the answer selection.

We analyzed the TREC-9 answer corpus and examined the output of Qanda's question processing component with respect to those questions. This indicated some mismatches between the system's expectations about answer types and the actual answers found in TREC-9. We hope to remedy these problems, as well as subject other system components to such scrutiny. We would also like to analyze the TREC-11 answers in a like manner.

References

Apache Software Foundation, 2002. "Jakarta Lucene—Overview". http://jakarta.apache.org/lucene/.

John D. Burger, John C. Henderson, William T. Morgan, 2002. "Statistical named entity recognizer adaptation", *Proceedings of the Conference on Natural Language Learning.* Taipei.

John D. Burger, Scott Mardis, 2002. "Qanda and the Catalyst architecture", *AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases.*

Krzysztof Czuba, John Prager, Jennifer Chu-Carroll, 2002. "A machine-learning approach to introspection in a question answering system", *Conference on Empirical Methods in Natural Language Processing.*

Simon Dik, Maria E. Hoffmann, Jan R. de Jong, Sei Ing Djiang, Harry Stroomer, Lourens de Vries, 1981. "On the typology of focus phenomena", in Teun Hoekstra, Harry van der Hulst, Michael Moortgat (eds.), *Perspectives on Functional Grammar.* Dordrecht: Foris.

Richard O. Duda, Peter E. Hart, David G. Stork, 2001. *Pattern Classification.* John Wiley & Sons.

Jonathan G. Fiscus, 1997. "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER)", In *Proceedings of the European Conference on Speech Technology,* volume 4.

Dan Gusfield, 1993. "Efficient methods for multiple sequence alignment with guaranteed error bounds", *Bulletin of Mathematical Biology,* 55(1).

Eva Hajicová, 1984. "Topic and focus." In Jan Horecky (ed.), *Contributions to Functional Syntax, Semantics, and Language Comprehension.*

John C. Henderson, 1999. *Exploiting Diversity for Natural Language Parsing.* Ph.D. thesis, Johns Hopkins University.

Eduard Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, Chin-Yew Lin, 2000. "Question answering in Webclopedia", In *Proceedings of the Ninth Text Retrieval Conference (TREC-9).*

Abraham Ittycheriah, Martin Franz, Salim Roukos, 2001. "IBM's statistical question answering system", *Proceedings of the Tenth Text Retrieval Conference (TREC-10).*

Beth Levin, 1993. *English Verb Classes and Alternations: A Preliminary Investigation.* University of Chicago Press.

Marc Light, Gideon S. Mann, Ellen Riloff, Eric Breck, 2001. "Analyses for elucidating current question answering technology", *Natural Language Engineering* 7(4).

Michael S. Rochemont, 1986. *Focus in Generative Grammar.* Amsterdam: John Benjamins.

Ralph Weischedel, Scott Miller, Ada Brunstein, Robert Granville, Jonathan May, Lance Ramshaw, 2002. "Answering questions through understanding and analysis (AQUA)". In notebook from *AQUAINT R&D Program Phase 1 Mid-Year Workshop,* Monterey CA, June.

TREC-11 Experiments at NII: The Effect of Virtual Relevant Documents in Batch Filtering

Kyung-Soon Lee, Kyo Kageura, Akiko Aizawa

NII (National Institute of Informatics)
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, 101-8430, Japan
{kslee, kyo, akiko}@nii.ac.jp
http://research.nii.ac.jp/~{kslee, kyo, akiko}

1 Introduction

In machine learning techniques, many researches have shown the effectiveness according to training examples by sampling from training set and incorporating prior knowledge into training set.

Researches on document retrieval, text categorization and routing have shown the effects of learning by sampling relevant documents or non-relevant document from training set. Allan et al. (1995) considered only the top K non-relevant documents, which is the same number of all known relevant documents in the training set to learn a routing query. This is motivated by the need to have a balance between the number of the relevant and the negative documents in Rocchio's learning. Singhal et al. (1997) selectively used the non-relevant documents that belong to a query's domain to learn the feedback query. Kwok and Grunfeld (1997) selected the best training subset of the relevant documents for creation of a feedback query based on genetic algorithm. Most sampling techniques in machine learning aim at the reducing the size of training set.

On the other hand, many machine learning applications on image recognition, image classification and character recognition have incorporated prior knowledge about the desired behavior of the system into training data. Prior knowledge is information for the learning which is available in addition to the training examples and makes it possible to generalize from the training examples to novel test examples (DeCoste and Schölkopf, 2002). For example, image recognition system uses new examples by small distortions of the input image such as translations, rotations, scaling; speech recognition system produces those by time distortions or pitch shifts. In 3D object recognition problem, Poggio and Vetter (1992) exploited appropriate transformations to generate new views from a single 2D view. In handwritten digit recognition, DeCoste and Schölkopf

(2002) added virtual examples generated by simply shifting the images by one pixel in the four principal directions to the training examples. In incorporating prior knowledge, the open issue is to what extent transformations can be safely applied during training since some distortions can lead to significantly worse errors (DeCoste and Schölkopf, 2002).

In TREC-11 batch filtering, we have incorporated prior knowledge called *virtual relevant documents* to training documents by combining each two relevant documents pair and giving distinct weight for co-occurring terms on assumption that they might be related to the topic. Support vector machine (SVM) was used to learn decision boundary for the artificially enlarged training documents.

2. Virtual Relevant Documents at Batch Filtering

Intuitively, a document produced by concatenating two relevant documents will be relevant to the topic since one large size of document can be divided into two documents while preserving the topic. And relevant documents will share terms which describe the topic. This characteristic has been used in feature selection. Therefore, prior knowledge generated by multiplying weights of a term which is co-occurring in each two relevant documents pair will provide new information about the decision boundary for classification.

2.1 Virtual Relevant Documents

A document is represented as a weight vector, *di* = <w_1, w_2,..., w_k, ..., w_n>. The weight is calculated by LogTF, IDF and cosine normalization.

A virtual relevant document (VRD) is generated by combining two relevant documents in training documents. For *n* relevant documents, n*(n-1)/2 documents are produced:

$$ {}_nC_2 = \frac{n \cdot (n-1)}{2 \cdot 1} \tag{1} $$

The weight of term which occurs in two relevant documents is calculated by multiplying two weights of a term of each vector. The weight of a VRD is calculated as follows:

$$ W_{vij_k} = W_{di_k} \cdot W_{dj_k} \tag{2} $$

where vij_k is the term *k* of a VRD, di_k and dj_k is the term *k* of relevant document *di* and *dj*, respectively. If the term *k* does not occur in one document of two relevant documents, the weight is assigned as minimum value instead of zero value and is multiplied to keep the term's existence. Finally, the weight vector of terms is normalized by cosine normalization.

The effect of VRD is that if two relevant documents do not have any sharing term, the resulting VRD become generalized vector of two documents. If two relevant documents share common terms, the resulting VRD would represent strong indicator of relevance to the topic for co-occurring terms. In case of general terms which are not related to the topic, they will have low value by idf in basic vector representation. Therefore, their effects would not be strong.

2.2 Support Vectors

Given training documents which include not only training documents but also VRDs, we have used support vector machine (Vapnik, 1995). Support vectors (SVs) are essential subset of relevant and non-relevant examples in training set. They represent the whole training examples. In test phase. SVs are used for determining on which side of the decision boundary.

Scholkopf et al. (1995) and Vapnik (1995) observed that the SV set contains all information necessary to solve a given classification task. In handwritten digit recognition task, DeCoste and Schölkopf (2002) showed that it is sufficient to generate virtual examples only from the support vectors.

3. Experiments

3.1 Experimental Procedure

In the runs (kNl111bfl and kNl111bf2) submitted to TREC-11 batch filtering, VRDs are generated from whole relevant documents in training set (VRDs_TRs). In additional experiments, VRDs are generated from support vector set obtained after training SVM for training set (VRDs_SVs). In this paper, we compare two incorporating methods for batch filtering.

We used SVM^{light} system (Joachims, 1999), and trained classifiers via radial-basis function (RBF) kernels and left all SVM^{light} options that affect learning as their default value.

3.2 Results

3.2.1 Submitted Runs

We have submitted two runs tagged kNl111bfl and kNl111bf2. In the submitted runs, VRDs were generated from whole training documents. In kNl111bfl, VRDs were generated by multiplying weights from two relevant documents, and subtracting terms in non-relevant documents from in relevant documents. It also included virtual non-relevant documents produced by averaging weights from two non-relevant documents, and subtracting terms in relevant documents from in non-relevant documents. In kNl111bf2, VRDs were generated by multiplying weights.

For evaluation measure T11U and T11F, refer TREC-11 filtering track guideline. For the assessor topic, the performances of kNII11bf1 and kNII11bf2 on MeanT11U are 0.305 and 0.302, respectively. The performances of kNII11bf1 and kNII11bf2 on MeanT11F are 0.190 and 0.188, respectively. The results of two runs are almost similar. It means that virtual non-relevant documents do not affect the performance.

3.2.2 Additional Experimental Results

We have compared the effectiveness for VRDs generated from different sources:

- Org training set: performance of SVM for training set.
- VRDs_SVs: the performance of SVM after incorporating prior knowledge generated from relevant support vector set into support vector set.

Table 1 shows the performance for assessor topics (from R101 to R150).

Table 1. The performances on original training set and incorporating VRDs into SV set.

Evaluation Measure	Org training set	VRDs_SVs	Performance change
Mean T11U	0.359	0.376	4.7%
Mean T11F	0.090	0.190	111.1%
Avg. Precision	0.269	0.400	48.7%
Avg. Recall	0.046	0.101	119.6%
Micro-avg. F1	0.181	0.310	71.3%

Table 2. The statistics of information used in support vector learning (RDs: relevant documents, NRDs: non-relevant documents).

	Org training set	VRDs_SVs
Avg # of training documents	861.48	328.24
Avg # of relevant documents	12.78	216.08
(Avg # of VRDs)	-	(204.92)
Avg # of SV set	123.32	119.44
(Avg # of SVs taken from VRDs)	-	(15.32)
(Avg # of SVs taken from RDs in SV set)	-	(10.64)
(Avg # of SVs taken from NRDs in SV set)	-	(93.48)

Table 2 shows the statistics of information in learning process for assessor topics. A lot of relevant SVs included in the new support vectors are taken from VRDs generated artificially, rather than original relevant documents. And the size of SV set learned from VRDS_SVs are similar with that learned from original training set.

In the experimental results, the proposed method achieved a significant performance improvement on the overall evaluation measures. These results indicate that our VRDs give new information to learn decision boundary in SVM. It is only 47 topics among the total 50 topics that VRDs_SVs improved pcrformance compared to Org training set. Therefore, VRDs generated by multiplying two relevant documents can be applied to transformation in batch filtering task.

4 Discussion

In TREC-11 batch filtering, we have incorporated virtual relevant documents to training documents by combining each two relevant documents pair on assumption that they might be related to the topic. Support vector machine was used to learn from the artificially enlarged training documents. By adding virtual relevant documents generated by transformation of original documents to training set, we could improve performance significantly. However, the base performance of SVM on the training set is low for TREC-11 test collection. For many topics. SVM system classified test documents as non-relevant to the many topics. For future work, VRDs can be applied in other classification model and adapted new virtual transformation.

References

Allan, J., Ballesteros, L., Callan, J., Croft, W., and Lu, Z. (1996) Recent experiments with INQUERY. In Proc. of thc Fourth Text REtrieval Conference (TREC-4).

DeCoste, D. and Schölkopf, B. (2002) Training invariant support vector machines. *Machine Learning 46(1)*, pp.161-190.

Joachims, T.(1999) Making large-scale support vector machine learning practical. In *Advances in Kernel Methods: Support Vector Machines* (Schölkopf et al., 1999), MIT Press.

Kwok, K. and Grunfeld, L. (1997) TREC-5 English and Chinese retrieval experiments using PIRCS. In the Proc. of the Fifth Text REtrieval Conference (TREC-5).

Poggio, T. and Vetter, T. (1992) Recognition and structure from one 2D model view: observations on prototypes, object classes and symmetries. A.I. Memo No. 1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology.

Schölkopf, B., Burges, C., and Vapnik, V. (1995) Extracting support data for a givcn task. In Proc. of the First International Conference on Knowledge Discovery & Data Mining, Menlo Park. AAAI Press.

Singhal, A., Mitra, M., and Buckley, C. (1997) Learning routing queries in a query zone. In *Proc. of the Twentieth ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 21-29.

Vapnick, V. (1995). *The Nature of Statistical Learning Theory*, Springer-Verlag, New York.

A machine learning approach for QA and Novelty Tracks: NTT system description

Hideto Kazawa, Tsutomu Hirao, Hideki Isozaki and Eisaku Maeda
NTT Communication Science Laboratories
Nippon Telegraph and Telephone Corporation
{kazawa,hirao,isozaki,maeda}@cslab.kecl.ntt.co.jp

1 A unified approach to QA and Novelty Tasks

In one sense, the goals of QA and Novelty tasks are the same: extracting small document parts which are relevant to users' queries. Additionally, the unit of extraction is almost always fixed in both tasks. For QA, an answer is a noun phrase in most cases, and for Novelty, a sentence is recognized as the basic information unit.

This observation leads us to the following unified approach to both QA and Novelty tasks: first identify information units in documents, then judge whether each unit is relevant to the query. This two step approach is amenable to machine learning methods because each step can be cast as a classification problem. For example, noun phrase identification can be achieved by classifying each word into the start/middle/end/exterior of a noun phrase; sentence identification by classifying whether each period marks the of a sentence. Additionally, relevance judgment can be regarded as the classification of a pair of query and an information unit into a relevant-pair or non-relevant-pair.

In QA and Novelty Tracks at TREC 2002, we studied the feasibility of this two step approach, using Support Vector Machines as the learning algorithm of the classifiers. Since many studies on identifying information units have already been reported, we concentrate on the relevance judgment step in QA and Novelty tasks in this paper.

2 Question Answering Track

Because of limited time, we applied our machine learning approach only to questions concerning dates and quantities: the other questions were processed in the same way as reported in [1]. Hereafter, we limit ourselves to the questions about dates and quantities.

2.1 Answer Candidate Identification

For date/quantity questions, answers are likely base noun phrases (base NPs), including date or number. Thus, we extracted base NPs including date/number expressions as information units (answer candidates). However, since identifying such base NPs is arguably rather easy work, we constructed an answer candidate identification module based on a Naive-Bayes classifier instead of SVMs [1].

2.2 Relevant Candidate Selection

To train SVMs on the relevance jugdement of candidates, we created a training dataset that consists of pairs of date/quantity questions and their answers.

As positive example pairs, we used the pairs of the past QA Track questions that ask dates/quantities, and the answers collected from the past judgment files using answer patterns. In addition, we randomly selected incorrect answer candidates from the judgement files, then combined them with the questions to obtain negative example pairs.

Each question-candidate pair is converted into a feature vector. The features consists of the following two types of features.

Keyword densities

Keyword density in 5/10/20-word Hanning windows centered on the candidate.

Combined features

All combinations of question and candidate features. The question features consist of:

- Wh-word in the question,
- Headword of wh-phrase in the question and its WordNet category,
- Keywords (nouns, verbs) in the question and its WordNet category.

Here, wh-words are who, when, where, and what, whereas wh-phrases are phrases including wh-words.

The candidate features consist of:

- Headword of the candidate and its WordNet category,
- Binary indicator of whether the candidate includes number/month/day expressions,
- The number of digits in the candidates.

3 Novelty Track

3.1 Relevant Sentence Extraction

We made a training data set that consists of query-sentence pairs whose relevance was judged by us. Our data includes 21 queries chosen from TREC topics, which are not used in the novelty track, and 4044 sentences chosen from past TREC results.

To apply SVM, we transformed each query-sentence pair into a feature vector. The features consist of:

[1] In general, SVMs show higher performance than Naive-Bayes classifiers. However, we prefer Naive-Bayes in this case because training Naive-Bayes classifiers is very fast.

- Sentence position normalized by the document length.
- Sentence length normalized by the longest sentence length in the same document.
- The sum of the weights of the sentence vector.
- Keyword density in the sentence. The keywords are terms in the description section of the query and the title section of the document.
- Cosine between the headline term vector and the sentence term vector.
- Cosine between the query term vector and the sentence term vector.
- Cosine between the query term vector and the document term vector.

Here, term vectors are commonly-used tfidf vectors. The inverse document frequency (idf) is calculated using all the TIPSTER document sets.

For the TREC 2002 Novelty Track, we trained SVMs with the quadratic kernel, $(x \cdot x' + 1)^2$, and the Gaussian kernel, $\exp(-a|x - x'|^2)$.

3.2 "New" Sentence Selection

For the TREC 2002 Novelty Track, it is not sufficient to merely judge the relevance of each information unit (sentence); it is required that only "new" sentences be reported at the end.

To select "new" sentences from relevant sentences, we used Marginal Relevance (MR) as the selection criteria. Originally. MR was proposed as a measure of "information" increased by the addition of a new document to the documents already selected[2].

$$\mathrm{MR}_{\mathcal{D}}(d) = \lambda \mathrm{rel}(d) - (1 - \lambda) \max_{d' \in \mathcal{D}} \mathrm{sim}(d, d'). \tag{1}$$

Here, d is the document whose contribution to information increase is to be measured, and $\mathcal{D}$ is the set of documents already selected. The function $\mathrm{rel}(d)$ is the relevance of d to the query. and $\mathrm{sim}(d, d')$ is the "similarity" between d and d'.

To apply MR to a new sentence selection in the Novelty task, we modified MR as follows.

- Sentences are regarded as (very) short "documents".
- For the relevance measure of sentences, we use the value of a discriminant function which is used in the relevant sentence extraction step.
- For the similarity measure, we use the number of the common words between the sentences divided by the average number of words in both sentences.

In the new sentence selection step, we repeat the selection of the sentence with the largest MR and add it to the selected sentence set until the largest MR is less than 0.2 [2]

[2] We set $\lambda = 0.7$.

	Quadratic	Gaussian
P*R(rel)	0.0694	0.0664
P*R(new)	0.0545	0.0477

Table 1: Results of Novelty Track

3.3 Results

We submitted two runs for Novelty Tracks: quadratic kernel SVMs were used in one run and Gaussian kernel SVMs in the other.

Table 1 shows "precision multiplied by recall" values of our submissions. Quadratic kernel SVMs perform slightly better than Gaussian kernel ones.

References

[1] H. Kazawa, H. Isozaki and E. Maeda, "NTT Question Answering System in TREC 2001," Proc. of TREC 2001, pp.415–422. (2001)

[2] J. Carbonell and J. Goldstein, "The Use of MMR, Diversity-Based Reranking for Reordering Document and Producing Summaries," Proc. of SIGIR-98, pp.335–336. (1998)

Some Similarity Computation Methods in Novelty Detection

Ming-Feng Tsai and Hsin-Hsi Chen
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
E-mail: mftsai@nlg.csie.ntu.edu.tw; hh_chen@csie.ntu.edu.tw

Abstract

In the novelty task, the amount of information of a sentence that can be used in similarity computation is the major challenging issue. Some sort of information expansion methods was introduced to tackle this problem. Our approach to relevance identification was to expand the information of a sentence with the context of this sentence using a sliding window method. The similarity was measured by the number of words of a topic description that match the sentences within a window. Besides, WordNet was employed to relax word match operation to inexact match. In the novelty detection part, we first applied a coherent text segmentation algorithm to partition the sentences extracted from the relevance identification part into several coherent segments denoting sub-topics. Then we compute the similarity of each sentence with each segment. A sentence was in terms of a sentence-segment similarity vector. Two sentences are regarded as similar if they are related to the same sub-topics. In this way, the redundant sentences were filtered out.

1 Introduction

Information explosion is one of challenging problems in the new information era. How to obtain relevant information from a large amount of data collection has become important. Current information retrieval (IR) systems only return documents satisfying users' information needs, but they do not locate the relevant sentences. Users have to go through the whole documents to find the relevant information. Moreover, traditional IR systems do not tell out which sentences contribute new information. To filter the redundant information and locate the novel information becomes more and more important for many emerging applications like summarization and question-answering. Novelty track, the new task of TREC, aims to locate relevant and new sentences (within context) rather than the whole documents containing duplicate and extraneous information.

Only few attempts have been made so far on novelty detection problem, because there is little agreement to the definition of *novelty* and the lack of evaluation data. In Topic Detection and Tracking (TDT) project (Allan, Carbonnell and Yamron, 2002), link detection task relates news stories on the same topic (Chen and Ku, 2002) and first story detection tries to find out the first article of a new event. It is some sort of novelty detection on document level. In novelty track of TREC, the basic unit that we confront with is a sentence. The amount of information of a sentence that can be used in similarity computation is the major challenging issue. In multi-document summarization (Chen and Huang, 1999; Chen and Lin, 2000), we faced the similar problem. We had to compute the similarity of meaningful units, which contain less information than passages and documents. Word matching and thesaurus expansion were adopted to tell out if two meaningful units touch on the same theme.

This paper shows how to extract relevant sentences from several known relevant documents, and how to determine new sentences from the extracted relevant sentences. The decision about what information is new depends on the order of the occurrence of the information. In other words, "a novel sentence" means that all of the relevant information in this sentence is never covered by the relevant sentences delivered previously. Section 2 presents the architecture of our system. It uses sliding window to exact relevant sentences and uses relevant segments to exact novel sentences. Section 3 shows the performance of this system and makes some discussions. Section 4 concludes the remarks.

2 Architecture

Figure 1 shows the architecture of our novelty system. It is composed of two major components, i.e., a relevance detector and a novelty detector. The relevance detector receives a sequence of sentences from known relevant documents, and determines which sentence is on topic. Those relevant sentences will be delivered to the novelty detector and the redundant sentences will be filtered out. The remaining sentences are *new (novel)* and *relevant*.

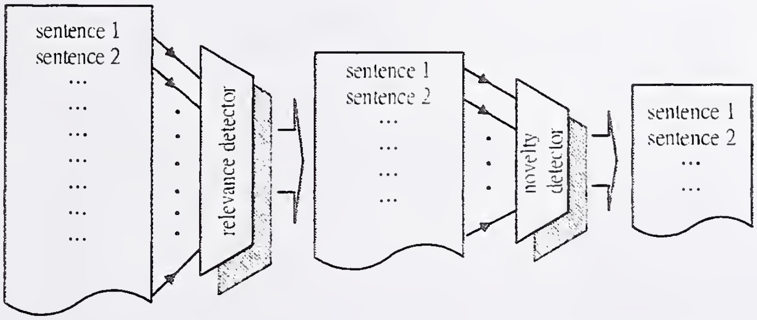


Figure 1: Architecture of Our Novelty System

The basic idea in our study is to measure the similarity of the sentences in the relevant documents. The following subsections will deal with the similarity model, relevance detector and novelty detector in sequence.

2.1 Similarity Model

Because the basic unit of similarity measure is a sentence instead of the whole document, we have to deal with the problem of less information in a sentence during distinguishing relevant and irrelevant sentences. Predicate-argument structure forms the kernel of a sentence, thus verbs and nouns are important features for similarity measures. All the sentences were parsed using Eric Brill's part-of-speech tagger. After tagging, nouns and verbs were extracted. Then we utilized WordNet to find the synonymous terms for inexact matching. Noun and verb taxonomies with hyponymy/hypernymy relations were consulted. The shortest path of each sense of word w_1 to each sense of word w_2, denoted $dist(w_1, w_2)$, was computed. Figure 2 demonstrates an example. Each note represents a synset in WordNet. In this example, the distance between *universe* and *sky* is 4.

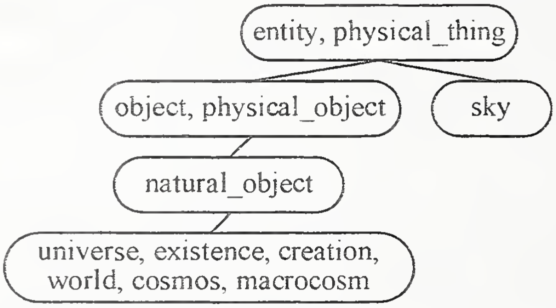


Figure 2: An Example of Distance Measurement

A threshold is employed to decide whether two words are similar or not. If their distance is less than the threshold, then 0.5 is added in the matching score. In summary, our similarity model is shown as follows:

- Nouns in one sentence are matched to nouns in another sentence, so are verbs. The value of 1 is added to the matching score for each exact matching.
- In inexact matching, we set word distance threshold to 4. In other words, if the $dist(w_1, w_2)$ is less than the threshold, the value of 0.5 is added to the matching score.
- Each term is matched only once.

The similarity of two sentences is in terms of noun-similarity and verb-similarity:

$$noun\_sim(s_1, s_2) = \frac{m}{\sqrt{ab}} \qquad (1)$$

$$verb\_sim(s_1, s_2) = \frac{n}{\sqrt{cd}} \qquad (2)$$

$$sim(s_1, s_2) = noun\_sim(s_1, s_2) + verb\_sim(s_1, s_2) \qquad (3)$$

where s_1 and s_2 denote two sentences, respectively;
m and n denote the number of matching nouns and verbs, respectively;
a and b are the total number of nouns in s_1 and s_2, respectively; and
c and d are the total number of verbs in s_1 and s_2, respectively.

2.2 Relevance Detector

The relevance detector aims to identify those sentences containing the relevant information from the known relevant documents. The approach to determine if a sentence is on topic is to use the above similarity function to measure the similarity of a sentence and the given topic. Its function is similar to traditional information retrieval system. The main difference is that the relevance detector extracts relevant information from sentences. The major problem of calculating similarity of a sentence and a topic is that sentence contains less information for comparison.

In Section 2.1, we try to augment a sentence with the synonymous terms retrieved from WordNet. We call it *within-sentence expansion*. Here one more expansion, called *between-sentence expansion* later, is considered. The context of a sentence is also a cue to determine relevance. In one extreme case, all the sentences surrounding the specific sentence form a context. But the context may be so large that noise may be introduced. In another extreme case, only the specific sentence is considered without adding any other sentence. In other words, it employs the information coming from itself. Trading the two extreme cases off, a sliding window controls how large a context is. Figure 2 shows a sliding window of size 2.

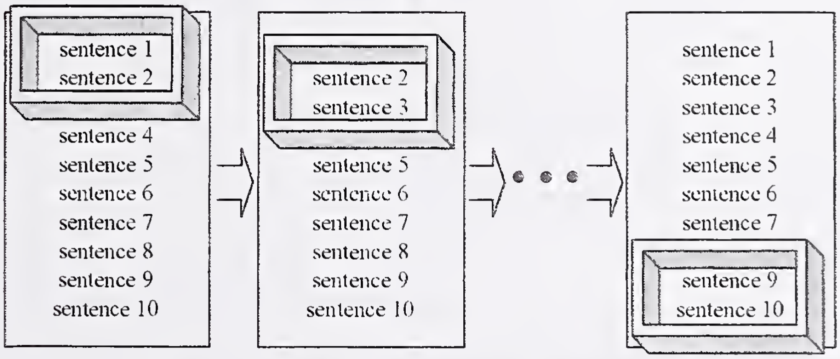


Figure 3: A Sliding Window (window size = 2)

A predefined relevance threshold, $TH_{relevance}$, is employed to determine whether sentences within a window are on topic or not. The sentences within a window are on topic if the similarity is larger than the predefined threshold. That is, if the sentences within a window are on topic, then those sentences within a window are identified as relevant sentences and sent to the next component, i.e., novelty detector. The window size and the relevance threshold, $TH_{relevance}$, are trained from the pre-released sample data.

2.3 Novelty Detector

The next step is to detect new information among the sentences extracted by the relevance detector. It would be better to say that we plan to filter the redundant sentences among the relevant sentences. The key issue on the detection of new information is how to differentiate the meaning of sentences accurately. Sentences may contain too less information to distinguish their differences, so that certain information expansion method is required.

We postulate that the relevant sentences may touch on several particular sub-topics. Under this postulation, a text segmentation algorithm developed by Utiyama, *et al.* (2001) was employed to partition the relevant sentences into several segments. Each segment corresponds to a sub-topic. This algorithm finds the maximum-probability coherent segmentation of a given text. The similarity between each sentence and each segment is calculated, and then each sentence is represented as a sentence-segment similarity vector. Two sentences are regarded as similar if they are related to the same sub-topics. In this way, the redundant sentences are filtered out and only the novel sentences are kept. Figure 4 sketches this idea.

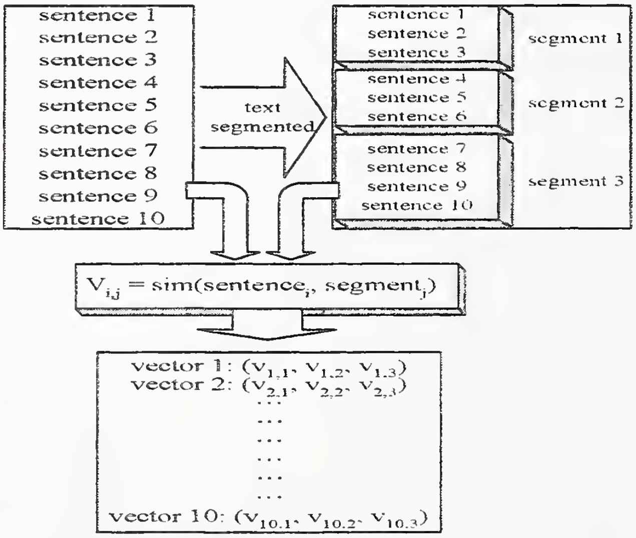

sentence 1
sentence 2
sentence 3
sentence 4
sentence 5
sentence 6
sentence 7
sentence 8
sentence 9
sentence 10
text segmented
segment 1
segment 2
segment 3
$V_{i,j} = sim(sentence_i, segment_j)$
vector 1: $(v_{1,1}, v_{1,2}, v_{1,3})$
vector 2: $(v_{2,1}, v_{2,2}, v_{2,3})$
...
vector 10: $(v_{10,1}, v_{10,2}, v_{10,3})$

Figure 4: An Illustration of Novelty Detector

Assume a sentence s_i is represented as a vector ($v_{i,1}$, $v_{i,2}$, …, $v_{i,n}$), where n is the number of segments. Cosine function shown in formula (4) measures the similarities of two vectors.

$$\cos(s_i, s_j) = \frac{\sum_{k=1}^{n} v_{i.k} \times v_{j.k}}{\|s_i\| \cdot \|s_j\|} \qquad (4)$$

This value indicates that how similar sentences s_i and s_j are. From the other point of view, the higher value indicates sentence s_i is somewhat redundant relative to sentence s_j. A threshold of novelty decision, $TH_{novelty}$, determines the degree of redundancy. If the similarity score of sentences s_i and s_j is larger than $TH_{novelty}$, then one of the two sentences has to be filtered out depending to their temporal order. The remaining sentences are the result of the novelty detector. The novelty threshold, $TH_{novelty}$, was trained from the pre-released sample data set.

In the above approach, we employed the relevant data itself to select the new information. Alternatively, we may use a reference corpus and regard each relevant sentence as a query to this corpus. An IR system may retrieve top n documents from the reference corpus for each relevant sentence. Each retrieved document is assigned a weight 1/r, where r is a rank of a retrieved document. In this way, a sentence is still represented as a vector. Cosine function measures the similarity of any two sentences, and the novel sentence is selected.

3 Experimental Results

Traditional precision and recall is counted to measure the performance of our novelty system and the product of precision and recall is also calculated for TREC measure. In the relevance part, we used description 1, description 2 as well as narrative part of the topic to retrieve relevant sentences. WordNet 1.7.1 was employed.

Tables 1 and 2 show our official runs at TREC 2002 Novelty Track. Our result of novelty part is not so good in this experiment, because the threshold, $TH_{novelty}$, is set to 0.97. This setting is according to the observation in pre-released sample set. The novelty sentence is ten percent of relevant sentences, thus we applied high novelty threshold to filter more sentences. After evaluation results were returned, we found that assessor also considered the sentence is novel if the sentence is relevant. Therefore, we applied higher novelty threshold in the latter unofficial experiments. In this way, two sentences should have much higher similarity to pass the threshold if they are similar. The lower the probability two sentences pass the threshold, the higher the probability both sentences are novel.

Table 1. Performance of Official Relevance Detection

	Relevance Part		
	Precision (P)	Recall (R)	P*R
ntu1	0.07	0.47	0.037
ntu2	0.07	0.47	0.033
ntu3	0.08	0.40	0.037

Table 2. Performance of Official Novelty Detection

	Novelty Part		
	Precision (P)	Recall (R)	P*R
ntu1	0.07	0.07	0.009
ntu2	0.06	0.07	0.008
ntu3	0.09	0.06	0.010

Our unofficial results are shown as follows. The set of sentences randomly selected from the target documents is regarded as a baseline model, its P*R score is 0.006. Table 3 lists the performance of relevance detector. The threshold for relevance detector is set to 0.4. Performance of the system (i.e., the P*R value) is improved as window size is increased from 1 to 4. When the window size is increased a little larger after the critical point, the performance starts to decline. The results show that larger window size may incorporate useful context information, but it may also select more irrelevant sentences.

Table 3. Performance of Relevance Detector ($TH_{relevance} = 0.4$)

Window size	Precision (P)	Recall (R)	P*R
1	0.137	0.211	0.029
2	0.094	0.393	0.037
3	0.080	0.474	0.038
4	0.077	0.532	**0.041**
5	0.069	0.565	0.039

We chose the best performance of relevance part to experiment with the next component, Novelty detector. The experimental result is shown in Table 4. In this experiment, the novelty thresholds are set to 0.98 and 0.99. Table 4 indicates that more sentences are filtered as $TH_{noviety}$ is lower. The experimental result shows that the performance of revised novelty detector is two times better than that of the original one in the formal run. However, the performance is still not comparable to the human assessors. The major reason is that the result of relevance detector contains irrelevant sentences, so novelty detector false identifies that those irrelevant sentences contain new information. As we mention before, the relevance part is the major difficulty to overcome in this task.

Table 4. Performance of Novelty Detector

Novelty Threshold	Precision (P)	Recall (R)	P*R
0.98	0.123	0.132	0.016
0.99	0.099	0.221	**0.022**

4 Conclusions and Future Work

In this paper, we proposed an approach to identify sentences that are novel and redundant as well as relevant and irrelevant. The method of matching keywords and related words in sentences may not be appropriate to the relevance part. We presented an information expansion approach to deal with this problem. We postulated that if two sentences have the similar meaning, then their behavior on information retrieval to a reference corpus (relevant sentence segments or an independent corpus) is similar. The current estimators for our approach should be improved, even though they sometimes work well on some topics. The syntactic and semantic analysis of sentences may help distinguish relevant sentence from target corpus.

To use a similarity function to measure if a sentence is on topic is similar to the function of an IR system. We may use a reference corpus, and regard a topic and a sentence as queries to this corpus. An IR system may retrieve top *n* documents from the reference corpus for these two queries. Each retrieved document is assigned a relevant weight by the IR system. In this way, a topic and a sentence can be in terms of two weighting vectors. Cosine furiction measures their similarity, and the sentence with similarity score larger than a threshold is selected. The issues behind this approach include the reference corpus, the IR system, the number of documents reported, the similarity threshold, and the number of relevant sentences extracted.

The reference corpus consulted should be large enough to cover different themes for references. In the first experiments, the document sets used in TREC-6 text collection were considered as a reference corpus. It consists of 556,077 documents. In the initial experiments, Smart system with the basic setting (i.e., *tf***idf* scheme without relevance feedback) was employed. It had average precision 0.1459 on the TREC topics 301-350.

We compute the Cosine of a topic vector T and a given sentence vector S_i ($1 \le i \le m$), where m denotes total number of the given sentences. Assume normal distribution with mean μ and standard deviation σ is adopted to specify the similarity distribution of the given sentences with a topic.

$$\mu = \frac{\sum_{i=1}^{m} \cos(T, S_i)}{m} \tag{5}$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{m} (\cos(T, S_i) - \mu)^2}{m}} \tag{6}$$

$$\cos(s_i, s_j) = \frac{\sum_{k=1}^{n} v_{i.k} \times v_{j.k}}{\|s_i\| \cdot \|s_j\|} \tag{7}$$

The percentage n denotes that top n percentages of the given sentences will be reported. Similarity thresholds ($\mathrm{TH}_{\mathrm{relevance}}$) shown as follows are determined by these percentages.

$$\mathrm{TH}_{\mathrm{relevance}} = \mu + z\sigma \tag{8}$$

$$\phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-y^2/2} dy = 1 - n \tag{9}$$

Even though the above dynamic approach has better performance, it is still "fixed percentage" for every topic. We consider further how to select "good" percentages for individual topics. Larkey *et al.* (2002) showed that only 5% of the sentences contained relevant materials for average topic. From their collection statistics (Larkey *et al.*, 2002), we used linear regression as follows to capture the relationship between total number of the given sentences and number of the relevant sentences.

$$n = 47.903 - 0.006x \tag{10}$$

where x is total number of given sentences, and n is the percentage.

After computing n using Formula (10), we derived z using Formula (9) and finally $\mathrm{TH}_{\mathrm{relevance}}$ using Formula (8). Table 5 summarizes experimental results. For different size of ranked document lists, the performance is more stable (i.e., between 0.71 and 0.81). The best average P×R is 0.081, i.e., 42.41% of human performance.

Table 5. Performance of Relevance Detection with Dynamic Percentages

doc-size	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
P	0.14	0.14	0.14	0.14	0.14	0.14	0.14	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.13	0.12
R	0.46	0.48	0.49	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.51	0.51	0.51	0.50
P×R	**.072**	**.077**	**.079**	**.080**	**.081**	**.078**	**.078**	**.076**	**.075**	**.075**	**.074**	**.074**	**.074**	**.074**	**.074**	**.071**

Figure 5 lists the performance of each topic when 45 documents were returned by IR system. Two dotted lines, i.e., one is human performance (0.191) and the other one is baseline performance (0.006), are provided for reference. Performance of our system in 8 topics (358, 364, 365, 368, 397, 414, 433 and 449) is competitive to that of human judge. In contrast, performance in 6 topics (305, 312, 315, 419, 420, and 432) is lower than that of random selection. The average P×R of the remaining 36 topics are below human performance, but better than that of baseline model.

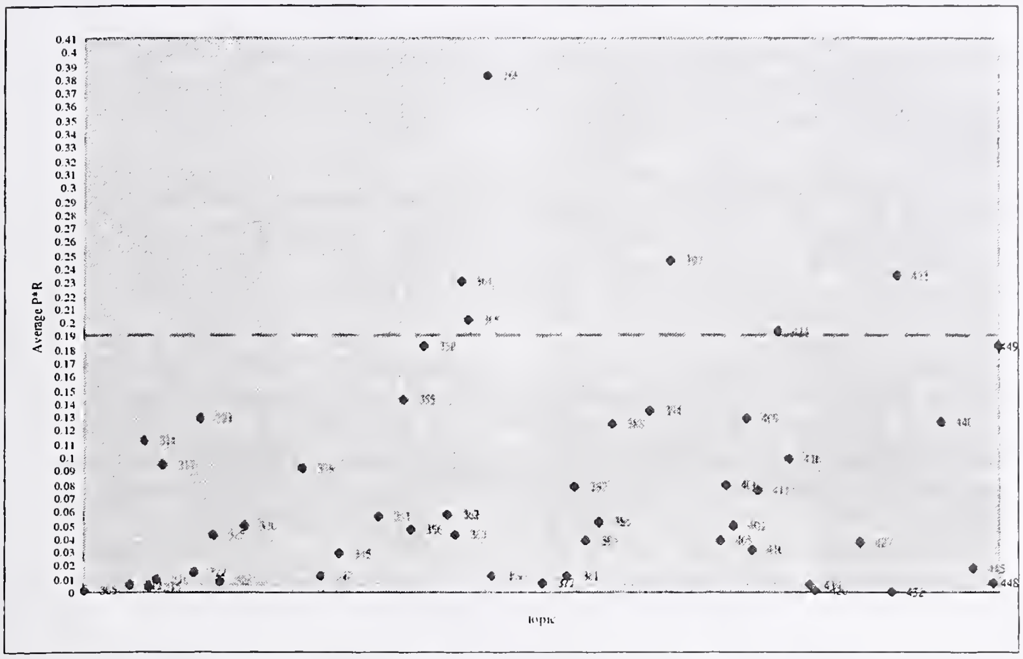


Figure 5. Average P×R of Relevance Identification for Each Topic

References

Allan, James; Carbonnell, Jaime; and Yamron, Jonathan (2002) *Topic Detection and Tracking: Event-Based Information Organization*, Kluwer, 2002.

Chen, Hsin-Hsi and Huang, Sheng-Jie (1999) "A Summarization System for Chinese News from Multiple Sources," *Proceedings of the 4th International Workshop on Information Retrieval with Asian Languages*, 1999, Taipei, Taiwan, pp. 1-7.

Chen, Hsin-Hsi and Ku, Lun-Wei (2002) "An NLP & IR Approach to Topic Detection," *Topic Detection and Tracking: Event-Based Information Organization*, James Allan, Jaime Carbonnell, Jonathan Yamron (Editors), Kluwer, pp. 243-264.

Chen, Hsin-Hsi and Lin, Chuan-Jie (2000) "A Multilingual News Summarizer," *Proceedings of 18th International Conference on Computational Linguistics*, 2000, University of Saarlandes, pp. 159-165.

Utiyama, M. and Isahara, H. (2001) "A statistical Model for Domain-Independent Text Segmentation," *Proceedings of ACL/EACL*, 2001, pp. 491-498.

The Integration of Lexical Knowledge and External Resources for Question Answering

Hui Yang and Tat-Seng Chua
School of Computing, National University of Singapore
Email: {yangh,chuats}@comp.nus.edu.sg

Abstract

For the short, factoid questions in TREC, the query terms we get from the original questions are either too brief or often do not contain most relevant information in the corpus. It will be very difficult to find the answer (especially exact answer) in a large text document collection because of the gap between the query space and the document space. In order to bridge this gap, there is a need to expand the original queries to include the terms in the document space. In this research, we investigate the integration of both the Web and WordNet in performing local context and lexical correlations to bridge the gap. In order to minimize the noise introduced by the external resources, we explore detailed question classes, fine-grained named entities, and successive constraint relaxation.

1. Introduction

We are participating in this year's Question Answering (QA) main task and it's our first time to take part in TREC. Question answering has recently received attention from many natural language processing communities [1][2][19]. Our goal is to retrieve the exact answers for the short, factoid questions in TREC. In our system, several modules have been developed. They are question processing, external resources adoption, document retrieval, candidate sentence selection and exact answer extraction.

During question parsing, the detailed question classes, answer types, original content query terms and NLP roles of the query terms are analyzed. We derive detailed question class ontology that corresponds to fine-grained named entities. This enables us to extract exact answer from the candidate sentences more accurately.

The original query terms can be used as the basis to locate potential answer candidates in the corpus. However, one major problem of doing this is that the query terms do not have sufficient coverage to locate most answer candidates. This is known as the semantic gap between the query space and document space. In order to bridge this gap, we use the knowledge of both the Web and lexical resources to expand the original query. We first use the original query to search the Web for top N web documents and then extract terms that co-occur frequently in the local context of the N-gram query terms. We next use WordNet to find other terms in the retrieved documents that are lexically related to the expanded query terms. The new query therefore contains terms that are related to the local context in the Web and the lexical context through WordNet. Finally, we use the expanded query to search for answer candidates through the MG system [20].

Candidate answer sentences are selected from the top returned documents and are ranked based on certain criteria to maximize the answer recall and precision. NL analysis is performed on these candidate sentences to extract POS, base Noun Phrases, Named Entities, etc. Answer selection is done by matching the expected answer type to the NL results. The nearest string with the expected answer type in the candidate sentence is returned as the final answer. Figure 1 gives the overview of our system architecture

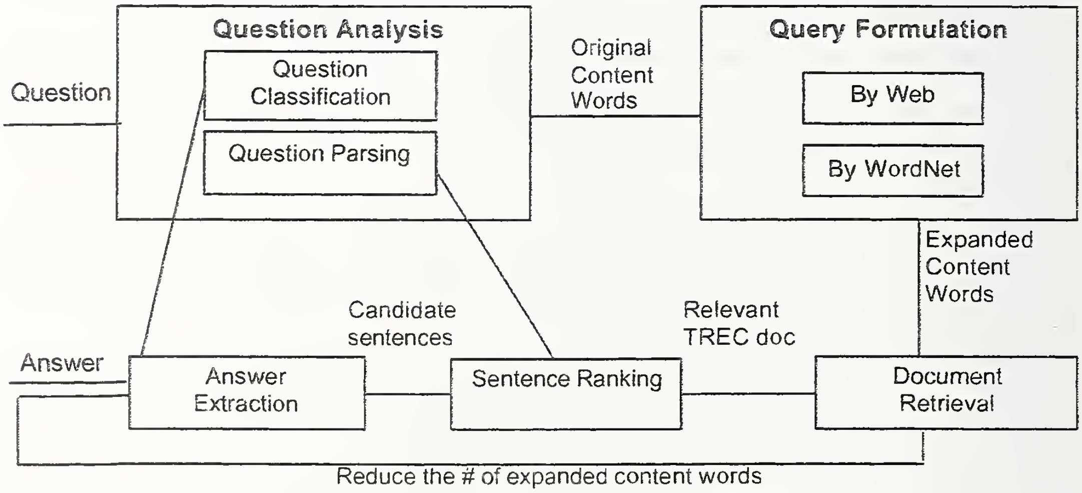


Figure 1: Overview of System Architecture

In this system, we focus on the techniques to expand the original query to locate most answer candidates. The resulting approach is efficient and has been found to be effective. Our experiments on TREC QA main task show good results when combining both local context and lexical information.

2. Question Processing

The purpose of our question processing is to find the specific nature of each question and to make full use of all the information in the question in order to find the best answer.

2.1. Question Classification

Question classification in our system is based on question focus and answer type. A rule-based question classifier is developed to determine the question focus and question class. There are seven main question classes in our system. They are: HUM (Human), LOC (Location), TME (Time), NUM (Number), OBJ (Object), DES (Description) and UNKNOWN (Unknown). The last type UNKNOWN is used to group questions that cannot be categorized into thc other classes. Different types of the questions are treated slightly differently in the following answer extraction module.

- Example 1: "Which city is the capital of Canada? " (Q-class: LOC)
- Example 2: "Which province is the capital of Canada in? " (Q-class: LOC)

Obviously, both of the questions belong to the type of *LOC (Location)* and their content words are almost the same, i.e., *capital and Canada*. However, they are expecting different answers, which should fall in different categories, i.e., city or state. The first question's answer will be *Ottawa* but the second's answer should be *Ontario*. In order to detect the subtle differences in the questions, we further classify the first 6 major question classes into 54 sub classes (see table 1 below). Under each main class, there is a special sub-class called *XXX_BASIC*, which is designed for questions that fall in the major class but do not suit any of the sub-classes. Our question classification is similar to the learning classifier developed by Li and Roth [13]. Currently, our rule -based classifier can reach an accuracy of over 98%.

Q-Class	Q-Sub-Class	#Trec11q	#Trec10q	Example
HUM	HUM_PERSON	43	19	Who is the governor of Colorado ?
	HUM_ORG	11	4	What car company invented the Edsel ?
	HUM_BASIC	41	35	Who is Tom Cruise married to ?
LOC	LOC_PLANET	1	2	Which planet did the spacecraft Magellan enable scientists to research extensively ?
	LOC_CITY	18	16	What is the capital city of Algeria ?
	LOC_CONTINENT	3	2	What continent is Scotland in ?
	LOC_COUNTRY	18	3	What country is Berlin in ?
	LOC_COUNTY	3	2	What county is Elmira , NY in ?
	LOC_STATE	3	4	Which state has the longest coastline on the Atlantic Ocean ?
	LOC_PROVINCE	2	2	What province is Calgary located in ?
	LOC_TOWN	2	0	The Hindenburg disaster took place in 1937 in which New Jersey town ?
	LOC_RIVER	3	3	What river is called "China 's Sorrow" ?
	LOC_LAKE	2	2	What is the deepest lake in the world ?
	LOC_MOUNTAIN	1	2	What is the name of the volcano that destroyed the ancient city of Pompeii ?
	LOC_OCEAN	2	1	What body of water does the Colorado River flow into ?
	LOC_ISLAND	3	1	What is the world 's second largest island ?
	LOC_BASIC	50	29	Where is Devil 's Tower ?
NUM	NUM_COUNT	11	12	How many chromosomes does a human zygote have ?
	NUM_PRICE	5	1	How much does it cost to register a car in New Hampshire ?
	NUM_PERCENT	4	8	What percent of the U.S . is African American ?
	NUM_DISTANCE	22	16	What is the height of the tallest redwood ?
	NUM_WEIGHT	0	2	What is the average weight of a Yellow Labrador ?
	NUM_DEGREE	3	7	What is the boiling point of water ?
	NUM_AGE	9	7	How old was Nolan Ryan when he retired ?
	NUM_RANGE	2	0	What is the range for the number of passengers a Boeing 747 airplane can carry ?
	NUM_SPEED	3	5	How fast does a cheetah run ?
	NUM_FREQUENCY	1	0	How often does the United States government conduct an official population census ?
	NUM_SIZE	2	1	What 's the capacity of the Superdome ?
	NUM_AREA	1	1	How much area does the Everglades cover ?
	NUM_BASIC	8	5	How much vitamin C should you take in a day ?
TME	TME_YEAR	24	15	What year was Alaska purchased ?
	TME_MONTH	2	0	In what month are the most babies born ?
	TME_DAY	8	9	What day did Neil Armstrong land on the moon ?
	TME_BASIC	65	23	When was the telegraph invented ?
OBJ	OBJ_CURRENCY	2	5	What is the currency used in China ?
	OBJ_MUSIC	8	2	What was Aaron Copland 's most famous piece of music ?
	OBJ_ANIMAL	5	9	What is the state bird of Alaska ?
	OBJ_PLANT	2	5	What is the major crop grown in Arizona ?
	OBJ_BREED	1	1	What breed was Roy Rogers ' horse Trigger ?

	OBJ_COLOR	2	9	What are the colors of the Italian flag ?
	OBJ_RELIGION	1	1	What is the chief religion for Peru ?
	OBJ_WAR	1	1	What war is connected with the book " Charge of the Light Brigade" ?
	OBJ_LANGUAGE	1	2	What language do they speak in New Caledonia ?
	OBJ_WORK	3	1	Which long Lewis Carroll poem was turned into a musical on the London stage ?
	OBJ_PROFESSION	3	2	What was William Shakespeare 's occupation before he began to write plays ?
	OBJ_ENTERTAIN	2	1	What TV series did Pierce Brosnan play in ?
	OBJ_GAME	2	1	What card game uses only 48 cards ?
	OBJ_BASIC	61	190	What is the chemical formula for sulphur dioxide ?
DES	DES_ABB	9	7	What does CPR stand for ?
	DES_MEANING	1	6	What does " E Pluribus Unum " mean ?
	DES_MANNER	5	4	How did Mahatma Gandhi die ?
	DES_REASON	1	4	What are hiccups caused by ?
	DES_BASIC	14	10	What do you call a baby sloth ?

Table 1 : Question Classes

2.2. Question Parsing

Besides question classes, some important information are also extracted when we parse the question. They are crucial for the later processes. Detailed analysis is performed here in order to get as much useful information as possible. There are several kinds of word groups we extract from the original question. They are:

(1) *Content Words:* These include nouns, adjectives, numbers, and some non-trivial verbs, which appear in the question string. Part of Speech tagging is performed before we select the content words. For example: "*What mythical Scottish town appears for one day every 100 years?*" , the content word vector will be $\underline{q}^{(0)}$: **(mythical, Scottish, town, appears, one, day, 100, years)**

(2) *Basic Noun Phrases:* we use noun phrase recognizer to identity all basic noun phrases appear in the question. For the above example, the noun phrase vector $\underline{n}$: (" **mythical Scottish town**")

(3) *Head of the First Noun Phrase:* It refers to the noun follows the question header (e.g. what, which, how, etc) and carries the main meaning of the question focus. It can be the next noun or the last word in the next noun phrase after the question header. For the above example $\underline{h}$: **(town)**. Usually, there is only one such head for each question sentence.

(4) *Quotation Words:* For some of the questions, quotations appear in the question string. They should be given special treatment. The string inside quotation marks usually is longer than a noun phrase, sometimes it could be a full sentence. For example, *" What Broadway musical is the song " The Story is Me " from ? "* The quotation word vector will be $\underline{u}$: **("The Story is Me ")**

3. Query Expansion

After question processing, we need to locate the relevant documents and sentences from the TREC corpus. The most common way is to apply information retrieval techniques to find the relevant documents and candidate answer sentences. For the short, factual questions in TREC, the query terms we get from the original questions are either too brief or do not fully cover the terms used in the corpus.

Given a short query, $\underline{q}^{(0)} = [q_1^{(0)} q_2^{(0)} \ldots q_k^{(0)}]$ usually with k<=4, the problem for retrieving all the documents relevant to $\underline{q}^{(0)}$ is that *the query does not contain most of the terms used in the document space to represent the same concept.* Thus there is thus a need to expand the original query to bridge the gap between the query space and document space.

We use general open resources to overcome this problem. The external general resources that can be readily used include the Web, WordNet, Knowledge bases, and Query Logs. Many groups working on QA have recently used the Web [3][4][5][6][8][9][12][18] and WordNet [7][10][11][16][17] as resources for question answering. In our system, we integrate the external resources to expand the query. The new query is then used to look for the relevant documents and sentences in the QA Text Collection.

3.1. Using Web as the Generalized External Resource

The Web is the most rapidly growing and complete knowledge resource in the world now. The terms in the relevant documents retrieved from the Web are likely to be similar or even the same as those in the QA Text Collection since they are both news articles.

Original content words in the question are passed to the online search engine, e.g. Google, to search for documents in the Web. The terms in the relevant web documents that are highly correlated with the original query terms will be considered as candidates to expand the context of the original query. The steps are:

a) Get original query $\underline{q}^{(0)} = (q_1^{(0)}, q_2^{(0)}, \ldots, q_k^{(0)})$.

b) For $\underline{q}^{(0)}$, retrieve the top N documents from the Web.

c) $\forall q_i^{(0)} \in \underline{q}^{(0)}$, extract $\underline{w}_i$, which contains non-trivial words in the same sentence or within *p* words away from $q_i^{(0)}$ in the retrieved web documents.

d) Rank all $w_{ik} \in \underline{w}_i$ by computing its probability of co-occurrence with $q_i^{(0)}$ as:

$$pr(w_{ik}) = \frac{d_s(w_{ik} \wedge q_i^{(0)})}{d_s(w_{ik} \vee q_i^{(0)})} \qquad (1)$$

where, $d_s(w_{ik} \wedge q_i^{(0)})$ is the number of instances that w_{ik} and $q_i^{(0)}$ appear together, and $d_s(w_{ik} \vee q_i^{(0)})$ is the number of instances that either w_{ik} or $q_i^{(0)}$ appears.

e) Merge all $\underline{w}_i$ to form $\underline{C}_q$ for $\underline{q}^{(0)}$. Therefore, $\underline{C}_q$ contains the list of words that are highly correlated with the original query from web documents.

3.2. Use WordNet as the Generalized External Resource

The Web can only provide us the words that occur frequently with the original query terms in the local context. It however, lacks information on lexical relationships between these terms. To overcome this problem. we look up WordNet to find words that are lexically related to the original query terms. The glosses, synonyms, and hypernyms are considered to be useful in relating words. In this work, we consider glosses and synonyms only to relate terms. For example, from the glosses,

- Definition of *plant*: A <u>living organism</u> lacking the power of locomotion
- Definition of *animal*: A <u>living organism</u>characterized by voluntary movement

The common concept here is ***living organism***, which will link concept ***plant*** to concept ***animal***.

From WordNet, we can find gloss words $\underline{G}_q$ and synset words $\underline{S}_q$ for $\underline{q}^{(0)}$. If we expand the query by appending all the terms in the glosses and synsets, it tends to be too general and contain too many terms out of context. In general, we need to restrict $\underline{G}_q$ and $\underline{S}_q$ to those terms found in the web documents, i.e., those found in $\underline{C}_q$. Thus we circumvent this problem by using gloss and synset relations to increase the weights of appropriate context terms $w_k \in \underline{C}_q$ by:

- if $w_k \in \underline{G}_q$, increase w_k by α
- if $w_k \in \underline{S}_q$, increase w_k by β, $(0<\beta<\alpha<1)$ *(2)*

The final weight for each term in $\underline{C}_q$ is normalized for ranking. The new query is formed as

$$\underline{q}^{(1)} = \underline{q}^{(0)} + \{\text{top } m \text{ terms from } \underline{C}_q \text{ whose weights are below the selection threshold}\} \qquad (3)$$

Currently, we plan to use the Semantic Perceptron Net approach [14] to derive semantic groups in $\underline{C}_q$. $\underline{G}_q$ and $\underline{S}_q$ in order to derive a structured approach to utilize external knowledge.

4. Document & Candidate Answer Sentence Retrieval

We use the MG tool [20] in our system to index the documents. We choose Boolean retrieval because of the short queries and the need to maximize precision. After performing Boolean retrieval by using $\underline{q}^{(1)}$ to retrieve the top M documents (M = 50), if $\underline{q}^{(1)}$ does not return sufficient number of relevant documents, we reduce the extra terms added and repeat the Boolean search. Therefore, we successively relax the constraints to ensure precision in document retrieval.

The sentence is chosen as the basic unit for processing in our system. After performing sentence boundary detection. we use the following criteria to rank the relevance of a sentence to the question: *(Recall from query processing, we extracted $\underline{q}^{(0)}$, $\underline{n}$, $\underline{h}$, $\underline{u}$).* For each Sentence $Sent_j$, we match it with

- quotation words: W_{uj} = % of term overlap between $\underline{u}$ and $Sent_j$
- noun phrases: W_{nj} = % of phrase overlap between $\underline{n}$ and $Sent_j$
- head of first noun phrase: $W_{hj} = 1$ if there is a match and 0 otherwise
- original content words: W_{cj} = % of term overlap between $\underline{q}^{(0)}$ and $Sent_j$
- expanded content words: W_{ej} = % of term overlap between $\underline{q}^{(1-0)}$ and $Sent_j$. where $q^{(1-0)} = \underline{q}^{(1)} - \underline{q}^{(0)}$

The final score for the sentence is $S_j = \sum a_i W_{ij}$, where $\sum a_i = 1$, $W_{ij} \in \{ W_{uj}, W_{nj}, W_{hj}, W_{cj}, W_{ej} \}$. The top *K* sentences are then selected as the candidate answer sentences based on S_j.

5. Answer Extraction

Finally we perform the tagging of fine-grained named entities [15] on the top *K* sentences extracted from the previous steps. From these sentences, we extract the string that matches the Question classes (answer target) as the answer. Once an answer is found within the top ith sentence. the system will terminate the search for the rest of *(K-i)* sentences. When there is more than one matching strings in a single sentence, we will choose the string that is nearest to the original query terms. For example: for question "*Where did Dr. King give his speech in Washington?*" , we get:

- Q-class: ***LOC_BASIC***
- *<LOC_BASIC WASHINGTON>* KING-DREAM _ *<LOC_BASIC WASHINGTON>* _ In the *<NUM_PERIOD 35 years>* since Dr . *<HUM_PERSON Martin Luther King>* Jr . delivered his `` I Have a Dream " speech at the *<LOC_BASIC Lincoln Memorial>* , how have economic and social conditions changed for *<LOC_CONTINENT African>* Americans ?

For question class ***LOC_BASIC***, we look for all the sub categories under LOC and we will get *WASHINGTON, WASHINGTON, Lincoln Memorial* and *African* as answer candidates. Among them, ***Lincoln Memorial*** is the nearest string to original content word speech, and hence is picked as the exact answer.

For some questions, we cannot find any answer. Our solution is to reduce the number of the expanded query terms and repeat the document/sentence retrieval and answer extraction process for up to m iterations (m=5). If we still cannot find an exact answer, NIL is returned as the answer. We call this method *successive constraint relaxation*, which helps to increase the recall while preserving precision.

6. Result Analysis

We answered 290 questions correctly with un-interpolated average precision of 0.61. Figure 2 shows that our system works well for most of the easy questions (right side of the figure), and has reasonable performance for the difficult ones.

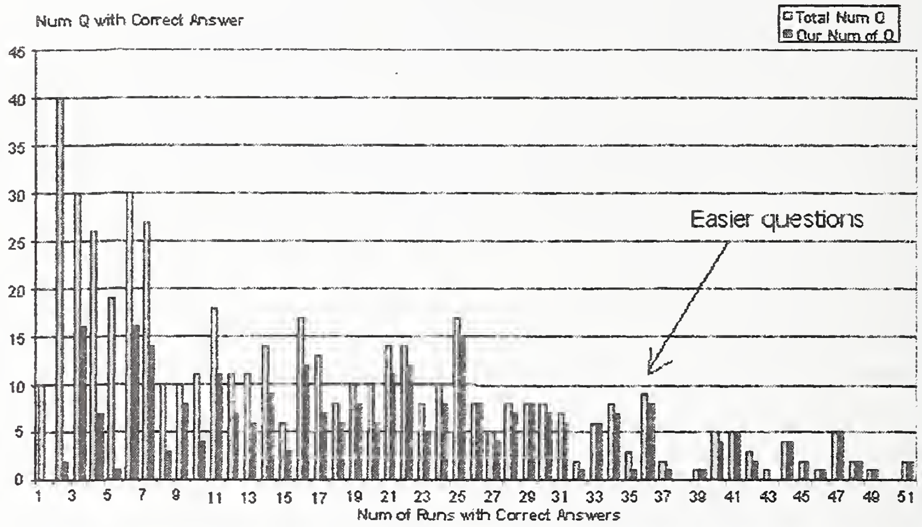


Figure 2: Question Difficulty Distribution

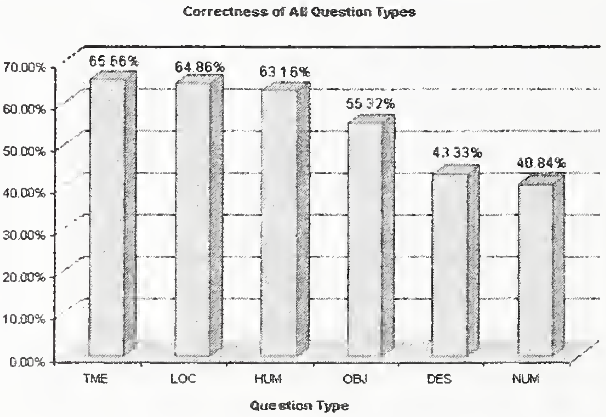


Figure 3: Answer Accuracy of All the Question Types

We also found that the accuracy of the exact answers differ for different type of questions (see Figure 3). For some question classes, like Time, Location and Human, our system gives quite high performance. For Description, Number and Object questions, we still need to find better techniques to improve the performance.

Another problem is that we have too many questions with NIL answers. The precision for recognizing NIL answer is low: 41 / 170 = 0.241, although the recall for NIL answer is satisfactory: 41 / 46 = 0.891. As a result, the overall system recall (consider both questions with non-NIL and those with NIL answer) is not satisfactory as compared to precision. This is because we use the boolean search to look for relevant TREC documents. Only the documents containing all the query terms are returned. This restriction might be too strict.

7. Future Work

We are currently refining our approach in several directions. First, we are refining our terms correlation by considering a combination of local context, global context and lexical correlations. Second, we are working towards a template-based approach on answer selection that incorporates some of the current ideas on question profiling and answer proofing, etc. Third, we will explore the structured use of external knowledge using the *semantic perceptron net* approach [14]. Our longer-term research plan includes Interactive QA, and the handling of more difficult analysis and opinion question types.

References

[1] AAAI Spring Symposium Series (2002). Mining Answers from Text and Knowledge Bases.

[2] ACL-EACL (2002). Workshop on Open-domain Question Answering.

[3] E. Agichtein, S. Lawrence and L. Gravano (2001). " Learning search engine specific query transformations for question answering" . In Proceedings of the 10th World Wide Web Conference (WWW10), 169-178.

[4] E.Brill,J.Lin,M.Banko,S.Dumais,and A.Ng, " Data-intensive question answering" , Text REtrieval Conference 2001

[5] E. Brill, S. Dumais and M. Banko (2002). "An analysis of the AskMSR question-answering system." In Proceedings of 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002).

[6] S. Buchholz (2002). " Using grammatical relations, answer frequencies and the World Wide Web for TREC question Answering" . In Proceedings of the Tenth Text Retrieval Conference (TREC 2001).

[7] J. Chen, A. R. Diekema, M.D. Taffet, N. McCracken, N. E. Ozgencil, O. Yilmazel, E. D. Liddy (2002). " Question answering: CNLP at the TREC-10 question answering track" . In Proceedings of the Tenth Text Retrieval Conference (TREC 2001).

[8] C. Clarke, G. Cormack and T. Lynam (2002). "Web reinforced question answering." In Proceedings of the Tenth Text REtrieval Conference (TREC 2001).

[9] C. Clarke, G. Cormack and T. Lynam (2001). " Exploiting redundancy in question answering" . In Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'2001), 358-365.

[10] S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus and P. Morarescu (2001). " FALCON: Boosting knowledge for question answering" . In Proceedings of the Ninth Text Retrieval Conference (TREC-9), 479-488.

[11] E. Hovy, U. Hermjakob and C. Lin (2002). "The use of external knowledge in factoid QA." In Proceedings of the Tenth Text REtrieval Conference (TREC 2001).

[12] C. Kwok, O. Etzioni and D. Weld (2001). " Scaling question answering to the Web." In Proceedings of the 10th World Wide Web Conference (WWW'10), 150--161.

[13] Xin Li and Dan Roth, " Learning Question Classifiers", In Proceedings of the 19th International Conference on Computational Linguistics, 2002

[14] J. Liu and T. S. Chua, " Building semantic perceptron net for topic spotting" , In Proceedings of 37th Meeting of Association of Computational Linguistics (ACL 2001), Toulouse, France, Jul 2001, pages 370-377

[15] Gideon S. Mann, "Fine-Grained Proper Noun Ontologies for Question Answering", SemaNet'02: Building and Using Semantic Networks, 2002

[16] M. A. Pasca and S. M. Harabagiu (2001). "High performance question/answering" . In Proceedings of the 24th AnnualInternational ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'2001), 366-374.

[17] J. Prager, E. Brown, A. Coden and D. Radev (2000). " Question answering by predictive annotation" . In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'2000), 184-191.

[18] D. R. Radev, H. Qi, Z. Zheng, S. Blair-Goldensohn, Z. Zhang, W. Fan and J. Prager (2001). "Mining the web for answers to natural language questions" . In Proceeding of the 2001 ACM CIKM: Tenth International Conference on Information and Knowledge Management, 143-150

[19] E.M.Voorhees. " Overview of the TREC 2001 Question Answering Track." In Proceedings of the Tenth Text REtrieval Conference (TREC 2001)

[20] I. Witten, A. Moffat, and T. Bell, "Managing Gigabytes" , Morgan Kaufmann, 1999.

Temporal Multi-Resolution Framework for Shot Boundary Detection and Keyframe Extraction

Chandrashekhara A, HuaMin Feng and Tat-Seng Chua
School of Computing, National University of Singapore
Email: {chuats,fenghm,chandra}@comp.nus.edu.sg

Abstract

Video shot boundary detection and keyframe extraction is an important step in many video-processing applications. We observe that video shot boundary is a multi-resolution edge phenomenon in the feature space. In this experiment, we expanded our previous temporal multi-resolution analysis (TMRA) work by introducing the new feature vector based on motion, incorporating functions to detect flash and camera/object motion, and selecting automatic thresholds for noise elimination based on the type of video. The framework is used to extract meaningful keyframes. Experiments show that our new system can detect and characterize both the abrupt (CUT) and gradual (GT) transitions effectively. It has good accuracy for both the detection of transitions as well as their boundaries.

1. Introduction

Due to the presence of many types of transitions and the wide varying lengths of GTs, the task of detecting the type and location of transitions in video is a complex task. In fact, the transition of video is not a single resolution phenomenon. For example, although longer GT can't be observed at a high temporal resolution, it is apparent at a low temporal resolution of the same video stream. Thus the detection of transitions of video shots is a temporal multi-resolution problem. Information across resolutions can also be used to detect as well as locate both the CUT and GT transition points. Since wavelet is well known for its ability to model sharp discontinuities and to process signals according to scales [1], we employ Canny-like B-Spline wavelets in this multi-resolution analysis. This work provides: a) an unified approach for CUT and GT detection; b) accurate location of gradual transition boundary; c) adaptive threshold value selection based on video variance within a sliding window; d) flash elimination by characterizing the phenomenon in multi resolution; e) motion elimination by computing quadratic similarity measures within the transition and its neighborhood; and f) keyframe extraction.

2. Basic Theory

2.1 Video Representation

We model the video according to the content of the video frames in the stream. The feature for representing the content of video frames could be of any type: color, shape, texture or motion. Thus video is modeled in a N-dimensional feature space of : (a) gray-level representation, (b) RGB value, and (c) Optic flow/motion vector value. The dimension of the space depends on the dimensionality of the chosen features. Since the color-histogram representation has been found to be useful for the video segmentation problem, we use the N-color histogram for each frame of video. Our experiment shows that the local histogram-based method has difficulties in improving both the recall and precision of the shot boundary detection at the same time. In addition it has difficulty locating precise boundary of GT due to the flash and camera/object motions. To overcome this problem we constructed a motion-based feature using the motion-vectors of MPEG compressed stream. Besides the use of these features, we also use derivatives to detect the transitions. The maximas of the first order derivative or zero crossing in the second order derivative will correspond to transition points. In this paper, the first order derivative was taken for easier implementation.

By empirically observing GTs that exists in most video streams, we find that different types of GTs exist like fade in/fade out, dissolve, wipe, morph etc. Moreover, the length of the transition may vary greatly too. Different shot transitions have different characteristics, so it is hard to use just one single feature and single algorithm to capture the characteristics of all kinds of shot transitions efficiently. Just as the assumptions most existing algorithm follows, one can clearly observe that the content between shots change much more than intra-shot change. However different types of shot transitions are observable at different scales in the feature space. Whatever the type or length of the transition, there will always be a change big enough that we can detect. The difference is only the resolution of our observation. For CUT, we could see the change both in a detailed observation (between two successive frames), or a coarse observation (across several frames), while GT only shows the change in a coarse observation. So the transition must be defined with respect to different resolutions. By viewing the video at multiple resolutions, the CUT and those GTs could be unified. The only difference is that GTs means boundaries of signal in low resolutions while CUT means in all the resolutions. By making this fundamental observation that a video shot boundary is a multi-resolution phenomenon, we can characterize the transitions with the following features: the scale of the transition, the strength of the transition, and the singularity of the transition point. We have developed a unique multi-resolution analysis technique to detect and characterize both the CUT & GT shot boundaries.

2.2 Applying Wavelet

The multi-resolution phenomenon has been widely studied in other areas, and wavelets provide a good mathematical basis for such an analysis. In the analysis, we need to construct a scale space. The Gaussian scale-space approach is widely adopted as the Gaussian function is the unique kernel, which satisfies the causality property as guaranteed by the scaling theorem: Because the first order derivative of the Gaussian function could be a mother wavelet, one can easily show that the sharper variation points of the signal corresponds to the local maxima of the wavelet transform. Thus a maxima detection of the wavelet transform is equivalent to boundary detection. If the mother wavelets is the Canny wavelet, which is the first order derivative of the Gaussian , then

$$\psi^{a}(x)=\frac{d\theta}{dx} \tag{1}$$

and the dilation at scale is

$$\psi_{s}^{a}(x)=\frac{1}{s}\psi^{a}(\frac{x}{s}) \tag{2}$$

When choosing a dyadic scale sequence 2^{j} , we get:

$$\psi_{2^{j}}^{a}(x)=\frac{1}{2^{j}}\psi^{a}(\frac{x}{2^{j}}) \tag{3}$$

The wavelet transform here is defined as:

$$W_{s}^{a}f(x)=f\times\frac{1}{s}\varphi^{a}\left(\frac{x}{s}\right)=\frac{1}{s}f\times\frac{d}{d\frac{x}{s}}\theta\left(\frac{x}{s}\right)=s\frac{d}{dx}[f\times\theta_{s}(x)] \tag{4}$$

From the right side of equation (4), we can see that the resulting output is a smoothed signal generated by a Gaussian filter that calculates the first order derivatives. It could be shown here that this wavelet transform is equivalent to smoothening the signal by applying the different scale Gaussian filters and then calculate the first order derivatives. The detailed derivation of Equations (1-5) can be found in [3]. The local maximas of the resulting signal will indicate where the transitions happen, and the magnitude of the maximas will show the strength of the transitions. Tracing the maximas in different resolutions is equivalent to finding transition points in different resolutions. In many cases, the presence of noise may result in maximas too. We distinguish the real transitions from the noise by examining the cross-resolution information. A real transition will still be a maxima in all resolutions. However, the noise may be lost or eroded in a lower resolution of the smooth function [3].

3. Implementation

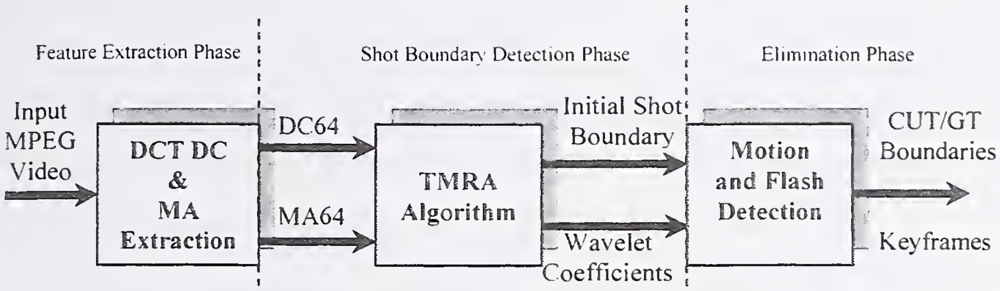


Figure 1. TMRA system for shot boundary detection

The TMRA system has 3 phases. In the feature extraction phase, feature vectors suitable for the TMRA are computed. In the shot boundary detection phase, transitions are characterized and the TMRA algorithm is applied to obtain the transition boundaries. This result will include many wrong transitions (insertions). In the elimination phase, motion analysis and flash detection are applied to remove insertions due to motion and abrupt flash noises. Figure 1 shows the system architecture for shot boundary detection. The following sections discuss in detail each of these three phases.

3.1 Feature extraction

We extract motion vector feature and the color histogram feature. The DC64 color histogram is computed by extracting the DCT DC value for each block in the frame. The value is quantized to 64 values. The MA64 direction histogram is computed using the motion vectors for each macro block and quantizing the angle values to 60 bins. Since the motion vectors tend to be sparse, a 3X3 median filtering is applied to the motion vectors. Also boundary blocks are not considered in the formation of the feature vectors, as they tend to be erroneous. The last 4 bins contain the macroblock type count of forward predicted, backward predicted, intra and skip macroblocks.

3.2 TMRA Algorithm

In this phase, TMRA algorithm is applied to determine the potential transition point and their type. It performs the wavelet transformation on the color and motion based feature vectors.

3.2.1 Locating potential transitions

The goal of video segmentation is not only to detect the occurrence of a transition, but also to locate the exact positions of the CUT/GT to segment the video. In a GT, both the start position and end position need to be detected. Low resolution of the wavelet coefficients helps to detect the occurrence; where as high resolution helps to characterize the start and end of the transition. In the higher resolution, the boundaries would show up as the maxima points. To identify this boundary we use both the DC64 and MA64 wavelet coefficients. For low resolution (Resolution 3) DC64 wavelet coefficients are used and for high resolution (Resolution 0) MA64 wavelet coefficients are used. The DC64 feature space fails to characterize the beginning and ending frames of the transitions accurately at the high resolution. This is due to the fact that the rate of change in DC64 feature space is not high and doesn't result in a distinguishable peak. Hence we designed a new feature space based on direction of motion along with counts representing static (skip) and intra (blocks having significant change) blocks. As a result of this we observe that even a gradual change resulted in an abrupt spike (peak) at the start and end of the transition. This is captured as the boundary of the transition. After we identify the local maxima points at some lower resolution (also called potential transitions), we use these local maxima points as the anchor points, and trace up to the higher resolution of the motion-based wavelet coefficients.

3.2.2 Adaptive thresholding

The problem of choosing appropriate threshold is a key issue in applying TMRA for shot boundary detection. Heuristically chosen global threshold is not suitable as the shot content changes from scene to scene. So adaptive thresholds are better than a simple global threshold. Here we use a sliding window method to calculate the thresholds. Our system has one weighting factor which can adaptively adjust based on the sliding window size and the standard deviation of dc64 feature of the neighborhood frames.. For different video clips, their standard deviations (see equation 5) are different. The standard deviation of home videos is larger than the general videos. The choice of sliding window size is also very important. In our system, we scan the whole wavelet coefficients of DC64 and choose the sliding window size as the sum of max interval of peak points and max interval of valley points (see equation 6). This removes most of the noise peak points due to brightness/contrast variations, blurring and small motions. The weighting factor can be used to adjust the threshold, which is chosen in the range of 0.8~1.5. In general, for home video, the weighting factor can be larger, in the range of 1.2~1.5, whereas for other video, it can be in the range of 0.8~1.1.

$$S = \sqrt{\frac{1}{N-2}\left(\sum_{i=1}^{N}(f_i - f_{i-1})^2 - \frac{\left(\sum_{i=1}^{N}|f_i - f_{i-1}|\right)^2}{N-1}\right)} \tag{5}$$

where N denotes the total frame number of the video sequence; f denotes the feature (DC64).

$$dis_p = \arg\max_{i \in N_p}\{D_i^{(p)}\}, dis_v = \arg\max_{i \in N_v}\{D_i^{(v)}\}$$
$$size = dis_p + dis_v \tag{6}$$

where Np denote the number of the peak points and Nv denote the number of valley points. $D_i^{(p)}$ is the interval of neighborhood peak points whereas $D_i^{(v)}$ is the interval of neighborhood valley points. dis_p and dis_v are maximum intervals of peak points and valley points, respectively.

3.3 Elimination and Keyframe Extraction

The last phase of the TMRA algorithm is the elimination of wrong transitions due to motion and abrupt flashes in the shot. Also representative keyframes are extracted for each shot. The following sections give a brief description on the method to characterize such activities in the multi resolution framework.

3.3.1 Flash detection

With our TMRA, flash is easily detected by testing the changes of frame's wavelet coefficients at all resolutions. For abrupt noise, the magnitude of coefficient value also decreases as the resolution decreases.

3.3.2 Camera/Object motion detection

With our ATMRA. we detect camera and object motion points. In principle, for the correct transitions (CUT/GT), the mean absolute differences (MAD) of DC64 and MA64 should be consistent. At CUT and GT points, the MADs are consistent across both the DC64 and MA64 feature space. If MAD changes are not consistent across DC64 and MA64 around the potential transition points, then it is likely to be a wrong transition caused by the camera/object motion.

Summarizing our motion detection method: first we compute three kinds of quadratic differences (distance between mean absolute difference of feature vector) for each potential transition we have found. They are represented as QMADbefore(similarity before the transition), QMADintra (similarity within the transition), QMADafter (similarity after the transition) as shown in Equations (7-9). Here $C_k^r = \dfrac{k!}{r!(k-r)!}$ is the normalizing factor. For each transition, we compute these three parameters for DC64 and MA64 feature, respectively.

$$QMAD_{before} = \left(\sum_{i=start-k}^{start-1} \sum_{j=i+1}^{start} | f_i - f_j | \right) / C_k^2 \tag{7}$$

$$QMAD_{inter} = \left(\sum_{i=start}^{end-1} \sum_{j=i+1}^{end} | f_i - f_j | \right) / C_{end-start+1}^2 \tag{8}$$

$$QMAD_{after} = \left(\sum_{i=end}^{end+k-1} \sum_{j=i+1}^{end+k} | f_i - f_j | \right) / C_k^2 \tag{9}$$

where: start and end represents the begin and end frame number of the potential transition. k represents the computing range ($2 \leq k \leq 9$). fi denotes the feature.

We remove those potential transitions that meet the following condition:

$$\left(\begin{array}{l} \left(QMAD_{inter}^{(dc)} \geq \left(QMAD_{before}^{(dc)} + QMAD_{after}^{(dc)}\right)/\ 2\right) \wedge \\ \left(QMAD_{inter}^{(mv)} < \left(QMAD_{before}^{(mv)} + QMAD_{after}^{(mv)}\right)/\ 2\right) \end{array} \right) \quad 10)$$

3.3.3 Keyframe extraction

Keyframes are very useful to summarize videos, and to provide access points into them. In this paper we also derive distinct keyframes to represent each shot. The keyframes are extracted as a by-product of multi-resolution analysis for shot boundary detection. We extend the concept of finding the scene transitions as local maxima's in the feature space; the local minima's can be chosen to represent the keyframe. Only the Resolution 3 (low resolution) is used to find the local minima points. For every shot two minima's are identifies, one in the DCT DC feature space and the other in the MA64 feature space. The color histogram distance between these two representative frames are computed and thresholded to chose either one or both the frames. Also a time constraint of 1 sec distance between the two keyframes is imposed. This ensures that the keyframes are distant and well represents the action in the video. The DCT DC selection results in color constant keyframes, where as the MA64 minima represents minimal motion angle change in the consecutive frames.

4. Experimental Results

The effectiveness of the algorithm was evaluated on TREC-2002 test data set. We submitted 2 runs represented as Nus1 and Nus2. The video contained a total of 2090 transitions. About 70% of them were cuts and 30 % gradual transitions and others. The results suffer from a poor detection of gradual transitions. We observed that our system throws many short gradual transitions (SGT) for single long gradual transitions. Also Fade-In-Out was considered as two separate transitions. We never eliminated the start and end transitions. By changing the SGT value to 4 we see that there is a 7% improvement in GT recall and 5% improvement in GT precision. Also we made small experiments to merge neighboring SGT's. The results show a 7% improvement in GT recall and 15 % improvement in Frame recall. These results are tabulated in the following table.

System Description	All		Cuts		Gradual			
	Rec	*Prec*	*Rec*	*Prec*	*Rec*	*Prec*	*F-Rec*	*F-Prec*
Nus1	0.621	0.615	0.742	0.670	0.313	0.411	0.301	0.833
Nus2	0.594	0.614	0.707	0.693	0.306	0.369	0.331	0.848
Nus1(with SGT=4)*	0.63	0.625	0.732	0.692	0.374	0.42	0.268	0.838
Nus1(With Merge)*	0.6	0.675	0.685	0.75	0.382	0.465	0.455	0.654

5. Conclusion

In conclusion, it has been demonstrated that TMRA framework offers a general and novel approach to flexibly and accurately probe the structure and content of digital video and meantime, it provides the ability to incorporate the new function to expand and improve the performance. Our future work is (1) to improve gradual transition detection and frame recall, (2) to investigate the active learning via artificial neural network to classify the CUT/GT, (3) to investigate the usage of other features for analyzing the video data, especially at semantic level, (4) to improve and to incorporate it into our video retrieval system.

Acknowledgements

The authors would like to acknowledge the support of the National Science and technology Board, and the Ministry of Education of Singapore for supporting this research under research grant RP3989903.

Reference

[1] Yu-Ping Wang and S.L.Lee.[1998]. "*Scale-Space Derived From B-Spline* ", PAMI Vol.20, No.10.

[2] Cohen A and Ryan R D, [1995]. *Wavelet and Multiscale Signal Processing*, Chapman and Hall Publishers.

[3] Y.Lin, M.S.Kankanhalli, and T.S.Chua,[2000] "*Temporal Multi-resolution Analysis for video Segmentation*", Proc. SPIE Conf. Storage and Retrieval for Media Database VIII, SPIE Vol. 3970, SPIE Press, Bellingham, Wash., Jan. 2000, pp.494-505.

Web based Pattern Mining and Matching Approach to Question Answering

Dell Zhang[1,2]
[1] Department of Computer Science
School of Computing
S15-05-24, 3 Science Drive 2
National University of Singapore
Singapore 117543
[2] Singapore-MIT Alliance
E4-04-10, 4 Engineering Drive 3
Singapore 117576
+65-68744251
dell.z@ieee.org

Wee Sun Lee[1,2]
[1] Department of Computer Science
School of Computing
S15-05-24, 3 Science Drive 2
National University of Singapore
Singapore 117543
[2] Singapore-MIT Alliance
E4-04-10, 4 Engineering Drive 3
Singapore 117576
+65-68744526
leews@comp.nus.edu.sg

Abstract

We describe herein a Web based pattern mining and matching approach to question answering. For each type of questions, a lot of textual patterns can be learned from the Web automatically, using the TREC QA track data as training examples. These textual patterns are assessed by the concepts of support and confidence, which are borrowed from the data mining community. Given a new unseen question, these textual patterns can be utilized to extract and rank the plausible answers on the Web. The performance of this approach has been evaluated also by the TREC QA track data.

1 Introduction

What a current information retrieval system or search engine can do is just "document retrieval", i.e., given some keywords it only returns the relevant documents that contain the keywords. However, what a user really wants is often a precise answer to a question. TREC has launched a QA track to support the competitive research on question answering, from 1999 (TREC8). The focus of TREC QA track is to build a fully automatic open-domain question answering system, which can answer factual questions based on very large document. Today, the TREC QA track [V0099][Voo00][Voo01] is the major large-scale evaluation environment for open-domain question answering systems.

Most of the recent question answering systems [V0099][Voo00][Voo01] require sophisticated linguistic knowledge or tools, such as parser, named-entity recognizer, ontology, WordNet, etc. However, at the TREC10 QA track [Voo01], the best performing system just used many textual patterns [SS01]. The power of textual patterns for question answering looks quite amazing and stimulating to us.

We describe herein a Web based pattern mining and matching approach to question answering. For each type of questions, a lot of textual patterns can be learned from the Web automatically, using the TREC QA track data as training examples. These textual patterns are assessed by the concepts of support and confidence, which are borrowed from the data mining community. Given a new unseen question, these textual patterns can be utilized to extract and rank the plausible answers on the Web. The performance of this approach has been evaluated also by the TREC QA track data.

To illustrate our approach, we would like to use the question "Who was the first American in space?" as a running sample in the following sections. This question was the No.21 test question in the TREC8 QA track.

2 System Overview

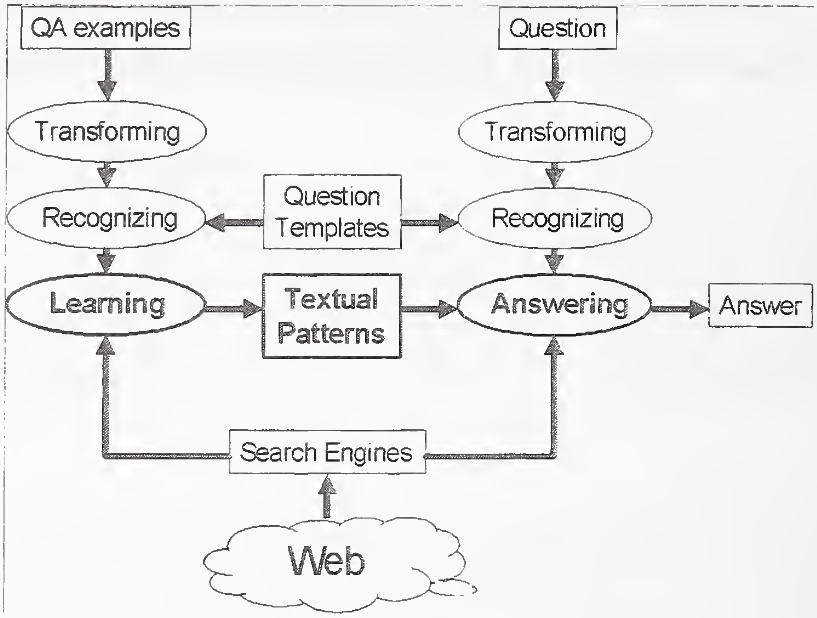


Figure 1, system architecture.

As shown in Figure 1, the system entails two main functions, one is learning and the other is answering. For both functions, the question has to be pre-processed by the transforming and recognizing module.

The answering part of the system relies on the textual patterns. A textual pattern can be in either of the following two forms.

_Q_ *<intermediate string>* _A_ *<boundary string>*
<boundary string> _A_ *<intermediate string>* _Q_

Here, _Q_ stands for the question key phrase and _A_ stands for the potential answer. The key phrase of a question is a continuous sequence of words in that question, which represents the primary object or event the question asking about. For instance, the key phrase of the sample question would be "the first American in space". A textual pattern actually describes the context of some potential answers to a specific class of questions.

Such textual patterns can be learned using some question-answer pairs as training examples. For instance, the correct answer to the sample question can be found in the string "In 1961, Alan Shepard became the first American in space ......", this observation suggests that the textual pattern ", _A_ became _Q_" can be used to answer similar questions like "Who was the first U.S. president?", "Who was the second man to walk on the moon?", and so on.

The learning part of the system will take advantage of the TREC QA track data as training examples. In each year's competition, TREC organizers issue several hundred questions to test the participant systems, and later release the regular expressions indicating the correct answers to each test question. Such TREC questions along with their answer regular expressions are our training examples.

3 Algorithms

3.1 Transforming

The transforming algorithm attempts to guess how the answer to the question may appear in a target sentence, i.e. a sentence that contains the answer, and then transforms the question to that format. We hope there will be more chances to find the answer after transforming the question. Currently two transforming methods are used.

For questions with an auxiliary do-verb and a main verb, the target sentence is likely to contain the verb in the conjugated form rather than separate verbs. For instance, the answer to the question "When did Nixon visit China?" would more likely to occur in the target sentence as "...... Nixon visited China ......" rather than "...... did Nixon visit China ......". So we transform the question by conjugating the auxiliary do-verb and the main verb. To locate the main verb like "visit" in the question, we have to parse the question using the MEI parser [Cha99]. To find the converted verb like "visited", we utilize PC-KIMMO [Ant90].

For questions with an auxiliary be-verb and a main verb (past particular), the target sentence is likely to contain these two verbs continuously together. For instance, the answer to the question "When was the telephone invented?" would more likely to occur in the target sentence as "......the telephone was invented ......" rather than "......was the telephone invented ......". So we transform the question by moving the auxiliary be-verb to the place just before the main verb. Again the MEI-parser is used to locate the main verb.

3.2 Recognizing

The recognizing algorithm determines the class of the question and the key phrase of the question. This was done by finding the appropriate template of the question.

Currently, we have defined 22 question classes, and each class has several templates formulated as regular expressions which indicate the possible appearance of this type questions. For instance, some of the templates in the ACRONYM class are as the following.

```
What is _Q_
What is the meaning of _Q_
What the (?:acronym|abbreviation) _Q_ (?:stands for|means)
What _Q_ (?:stands for|means)
What the initials _Q_ (?:stand for|mean)
_Q_ is (?:an|the) (?:acronym|abbreviation) for what
_Q_ stands for what
```

3.3 Learning

Assume the set of QA examples is E. Each QA example $e_i = (q_i, rea_i)$ consists of two parts, the question q_i and the regular expression of its correct answer rea_i. The following algorithms can learn the textual patterns of a specific question class c from the Web.

3.3.1 Construct Snippet Database

Given the QA examples of class c, $E^{(c)} = \{(q_i, rea_i) \mid (q_i, rea_i) \in E \wedge class(q_i) = c\}$, the following algorithm can automatically construct the snippet database of class c, $S^{(c)}$.

```
For each $(q_i, rea_i) \in E^{(c)}$, do {
        Submit $kp(q_i)$ as a phrase along with the words in $q_i$ to a Web search engine, such as Google
        (http://www.google.com/).
        Grab the search result returned by the search engine.
        Extract the text snippets from the search result.
        For each snippet $s_{ij}$, do {
                Insert a special symbol "_<_" at the head of $s_{ij}$.
                Append a special symbol "_>_" at the tail of $s_{ij}$.
                Replace every $kp(q_i)$ with a special symbol "_Q_".
                Find the answer $a_i$ using the answer regular expression $rea_i$.
                Replace every $a_i$ with a special symbol "_A_".
                Add $s_{ij}$ to $S^{(c)}$.
        }
}
```

For instance, the sample question "Who was the first American in space?" and its answer regular expression "((Alan (B\.)?)?Shephard)" can be taken as a QA example of the WHO-IS class. The following query will be submitted to Google, and then get the search result.

"the first American in space" Who was the first American in space?

A small portation of the snippet database of the WHO-IS class is shown in Figure 2.

```
......
_<_ in mind that Alan Shepard was _Q_, not the first man. _>_
_<_ John Glenn is not picked to be _Q_. _>_
_<_ Alan Shepard becomes _Q_. _>_
_<_ Then on 5 May 1961, less than a month after the Gagarin mission, Alan Shepard became _Q_.
_>_
_<_ with general relativity theory to follow, in 1905 60 Yuri Gagarin became the first man in space
in 1961 66 Alan Shepard became _Q_ in Yahoo! _>_
_<_ _Q_ was Alan Shepard, launched on May 5th Sea and Sky: Space Exploration 1961 - 1970. _>_
......
```

Figure 2, sample snippet database

Two special symbols "_<_" and "_>_" are used to simplify the algorithm, where "_<_" stands for the head of the snippet and "_>_" stands for the tail of the snippet.

3.3.2 Discover Textual Patterns

Given the snippet database of class c, $S^{(c)}$, the following algorithm can automatically discover the textual patterns of class c, $P^{(c)}$.

```
For each $s_{ij} \in S^{(c)}$, do {
        If ( $s_{ij}$ contains both "_Q_" and "_A_") then {
                Extract the textual pattern $p_k$ from the snippet using the following two regular
                expressions:
                        (\b_Q_\b(.*?)\b_A_\b\s*(_>_|\W|(\w+)))
                        ((_<_|\W|(\w+))\s*\b_A_\b(.*?)\b_Q_\b)
                If ( $p_k$ is not in $P^{(c)}$ yet) then Add $p_k$ to $P^{(c)}$.
        }
}
```

Some of the discovered textual patterns of the WHO-IS class is shown in Figure 3.

```
......
, _A_ became _Q_
_<_ _A_ was _Q_
_Q_ was _A_,
_A_ made history as _Q_
by _A_ ( _Q_
_Q_ , _A_ _>_
......
```

Figure 3, sample discovered textual patterns

3.3.3 Assess Textual Patterns

Given the textual patterns of class c, $P^{(c)}$, and the snippet database of class c, $S^{(c)}$, the following algorithm can automatically assess the textual patterns in $P^{(c)}$.

```
For each textual pattern $p_k \in P^{(c)}$, do {
        Translate the textual pattern $p_k$ into a regular expression $re(p_k)$, the special symbol "_A_" are replaced by "(.*?)", means this part can be matched by any string.
        Search $re(p_k)$ in $S^{(c)}$.
        Let $X$ denote the set of snippets which can match $re(p_k)$.
        Let $Y$ denote the set of snippets which can not only match $re(p_k)$, but also the string corresponding to the "(.*?)" part is just "_A_ ".
        $\text{support}(p_k) = \frac{|Y|}{|S^{(c)}|}$, $\text{confidence}(p_k) = \frac{|Y|}{|X|}$
        If $\text{support}(p_k)$ is less than the threshold $t_{\text{support}}$, then remove $p_k$ from $P^{(c)}$.
        If $\text{confidence}(p_k)$ is less than the threshold $t_{\text{confidence}}$, then remove $p_k$ from $P^{(c)}$.
}
```

In fact, a textual pattern can be considered as an association rule "context => answer". The concepts support and confidence are borrowed from the data mining community.

Some of the assessed textual patterns of the WHO-IS class is shown in Figure 4.

......	*confidence*
, _A_ became _Q_	0.09
_<_ _A_ was _Q_	0.11
_Q_ was _A_,	0.05
_<_ _A_ made history as _Q_	1.00
by _A_ (_Q_	0.66
_Q_ , _A_ _>_	0.14
......	

Figure 4, sample assessed textual patterns

3.4 Answering

Having the assessed textual patterns of each question, the following algorithm can be employed to answer a new unseen question.

For a new question q_{new}, determine its class c and its key phrase $kp(q_{new})$, by transforming and recognizing algorithms.

```
Submit $kp(q_{new})$ as a phrase along with the words in $q_{new}$ to a Web search engine, such as Google (http://www.google.com/).
Grab the search result returned by the search engine.
Extract the text snippets from the search result.
For each snippet $s_j$, do {
    Insert a special symbol "_<_" at the head of $s_j$.
    Append a special symbol "_>_" at the tail of $s_j$.
    Replace every $kp(q_{new})$ with a special symbol "_Q_".
    For each textual pattern $p_k \in P^{(c)}$, do {
        Translate the textual pattern $p_k$ into a regular expression $re(p_k)$, the special symbol "_A_" are replaced by "(.*?)", means this part can be matched by any string.
        If $s_j$ can match $re(p_k)$, then {
            Take the string corresponding to the "(.*?)" part as a plausible answer $a_{jk}$.
            If ($a_{jk}$ is not in $A^{(q)}$ yet) then {
                Add $a_{jk}$ to $A^{(q)}$
                Let $\text{confidence}(a_{jk}) = \text{confidence}(p_k)$
            }
            else {
                Increase $\text{confidence}(a_{jk})$ by $\text{confidence}(p_k)$
            }
        }
    }
}
Remove the unreasonable answers in $A^{(q)}$ using the stop-answers list and class-specific filters.
Sort all the found answers in $A^{(q)}$ by their confidence value.
Return the ordered top-N answers in $A^{(q)}$.
```

For instance, the answer to the sample question can be extracted from the snippet "Alan Shepard was the first American in space ......" with confidence 0.11 using a textual pattern of the WHO-IS class "_<_ _A_ was _Q_".

The list of stop-answers contains the strings which have no chance to be a correct answer in our opinion, such as "he", "today", "http", etc. And for each class of questions, we apply a class-specific filter which can help to remove the answers not for this class, e.g., a date class filter rejects a location string even it matches the textual pattern.

4 Experiments

Several experiments have been done to evaluate the performance of this approach, using the data from TREC8, TREC9 and TREC10. The questions with typo mistakes, the definition style questions like "Who is Colin Powell?", the questions which are syntactic rewrites of earlier questions (TREC9 test questions No.701-893), and the questions with no associated answer regular expressions have been removed from the data set. Note all the Web search results were retrieved from Google in the period July -- August 2002.

Test Data	*t#*	*Train Data*	*e#*	*r#*	*c#*	*MRR_all*	*MRR_ret*
TREC8	196	TREC9,10	757	93	52	0.22	0.46
TREC9	438	TREC10,8	515	282	155	0.27	0.42
TREC10	319	TREC8,9	634	230	120	0.28	0.39
TREC8,9,10	953	TREC8,9,10	953	634	509	0.53	0.79

Table 1, the performance of this approach on TREC8, TREC9 and TREC10 questions.

The experiment results are shown in Table 1. Here *t#* means the number of test questions, *e#* means the number of training examples, *r#* represents the how many questions the system has returned some answers for, *c#* represents how many questions the system has correctly answered. The MRR (Mean Reciprocal Rank) metric was used in TREC8, TREC9 and TREC10. Here *MRR_all* represents the MRR score over all test questions, while *MRR_ret* represents the MRR score over the questions which the system has returned some answers for.

The MRR score of this approach is not as high as that of the best question answering system in TREC. This discrepancy is due to many reasons. One important factor is that the answer regular expressions provided by TREC are quite limited, many correct answers such as "Alan B. Shepard, Jr." are judged wrong since they do not occur in the TREC specified document collection. Another interesting issue is time, the correct answers to some questions like "Who is the U.S. president?" will change over time. The Web is also messier than the TREC dccument collection.

This approach works quite well on simple questions, e.g. questions about ACRONYM, AUTHOR, BIRTHDATE, etc. The overall MRR score (MRR-all) for AUTHOR questions in TREC8 are above 0.55, while using the AUTHOR questions and answers in TREC9 and TREC10 as training examples. The performance of LAMP will dramatically drop down when the length of the question becomes longer, for instance, a TREC8 question averagely contains 9.93 words, but a correctly answered TREC8 question averagely contains 6.75 words.

The performance of this approach on TREC11 questions is as follows.

CWS (Confidence Weighted Score): 0.458
Precision of recognizing no answer is 39 / 293 = 0.133
Recall of recognizing no answer is 39 / 46 = 0.848

The above experiment results imply that this approach has high precision but low recall, i.e., this approach prefers "no answer" rather than "wrong answer". We think this property is good because "wrong answer" is usually worse than "no answer". And, this property allows this approach to be easily augmented with other approaches.

To increase recall for our TREC 11 entry, we augmented this approach with a simple answer extractor based on Support Vector Machines (SVM) [CS00] trained using features constructed from words in the question, words in the candidate sentence and the POS (Part-of-Speech) tags of words neighboring a candidate exact answer. To find a supporting document, we return the highest ranked document (in the list returned by the organizers) that contains the answer returned by the system.

The performance of the hybrid system on TREC 11 questions is as follows.

CWS (Confidence Weighted Score): 0.396
Precision of recognizing no answer is 0 / 2 = 0.000
Recall of recognizing no answer is 0 / 46 = 0.000

5 Conclusion

This approach distinguishes itself by its simplicity. It just uses the snippets in the Web search results, since it is time-consuming to download and analyze the original web documents. It does not require any sophisticated natural language processing on snippets, and does not need any advanced data structure, just simple hash table is enough. Because of this simplicity, LAMP is very efficient, which makes it perfect for online question answering.

One limitation of this approach is that one textual pattern can include only one question key phrase in the current stage. It does not work for the questions having multiple key phrases, possibly apart from each other. For example, to answer the question "How many calories are there in a Big Mac?", it would be better to use two question key phrases, "calories" and "Big Mac". Another drawback is that the textual patterns cannot handle long-distance relationships between the question key phrase and the answer. For example, the textual pattern "_Q_ became _A_," cannot locate the answer in the text snippet "Alan Shepard, who in 1961 became the first American in space and ......". However, the abundance and variation of the Web information makes it feasible to find answers on the Web with high probability through this approach, because the factual knowledge is usually replicated across the Web, expressed in many different forms [BLB+01].

References

[Ant90] E. L. Antworth. *PC-KIMMO: a two-level processor for morphological analysis.* Dallas, TX: Summer Institute of Linguistics, 1990.

[Cha99] E. Charniak. *A Maximum-Entropy-Inspired Parser.* Technical Report CS-99-12, Brown University, Computer Science Department, August 1999.

[CS00] C. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines.* Cambridge University Press, Cambridge, UK, 2000.

[BLB+01] E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data-Intensive Question Answering. In In *Proceedings of the 10th Text Retrieval Conference (TREC10)*,, pp.183-189, NIST, Gaithersburg, MD, 183- 189, 2001.

[SS01] M. M. Soubbotin and S. M. Soubbotin. Patterns of Potential Answer Expressions as Clues to the Right Answer. In *Proceedings of the 10th Text Retrieval Conference (TREC10)*, pp. 175- 182, NIST, Gaithersburg, MD, 2001.

[Voo99] E. Voorhees. The TREC-8 Question Answering Track Report. In *Proceedings of the 8th Text Retrieval Conference (TREC8)*, pp. 77-82, NIST, Gaithersburg, MD, 1999.

[Voo00] E. Voorhees. Overview of the TREC-9 Question Answering Track. In *Proceedings of the 9th Text Retrieval Conference (TREC9)*, pp. 71-80, NIST, Gaithersburg, MD, 2000.

[Voo01] E. Voorhees. Overview of the TREC 2001 Question Answering Track. In *Proceedings of the 10th Text Retrieval Conference (TREC10)*, pp. 157-165, NIST, Gaithersburg, MD, 2001.

More Statistical Power Needed: The OHSU TREC 2002 Interactive Track Experiments

William Hersh
Susan Moy
Dale Kraemer
Lynetta Sacherek
Daniel Olson

{hersh, moys, kraemerd, sacherek, olsondan }@ohsu.edu
Division of Medical Informatics & Outcomes Research
Oregon Health & Science University
Portland, OR, USA

The original goal for the Oregon Health & Science University (OHSU) TREC 2002 Interactive Track experiments was to perform some preliminary experiments comparing searching on tablet devices versus ordinary personal computers. Unfortunately, the vendor who had promised the devices we planned to use was unable to deliver them in time for the experiments. We therefore shifted our experimental focus to assessing user factors found in previous experiments to be associated with success, with a particular desire to assess the role of spatial visualization.

A variety of studies have demonstrated that spatial visualization is associated with successful computer use. Egan and Gomez have shown that spatial visualization is associated with two processes in text editing: finding the location of characters to be edited and generating a syntactically correct sequence of actions to complete the task [1]. Similarly, Vincente et al. have found that the ability to use a hierarchical file system is associated with spatial visualization as well as vocabulary skills [2]. In addition, Allen has demonstrated that this trait is associated with the appropriate selection of key words in searching [3]. We have previously found that the ability of medical and nurse practitioner students to answer clinical questions found spatial visualization to be highly predictive of success [4]. (Spatial visualization actually demonstrated multicollinearity with whether a searcher was a medical or nurse practitioner student, which may have been the actual predictive factor.) In previous TREC Interactive Track experiments, our results showed a trend towards spatial visualization being predictive of searching success in instance recall tasks, although they did not achieve statistical significance [5].

Methods

Our methods employed the consensus approach agreed upon by track participants and posted on the track Web site (http://www-nlpir.nist.gov/projects/t11i/guidelines.html). We used the .GOV Web collection created for the TREC 2002 Web Track as the searching data. The collection was accessed by using the Panoptic search engine. We followed the experimental protocol developed for the TREC-9 Interactive Track, which was designed to allow the comparison of two systems or system variants using a minimum of 16 searchers. Each searcher performed eight tasks, which are listed in Table 1.

Table 1 - Eight searching tasks in TREC 2002 Interactive Track.

1. You are traveling from the Netherlands, and want to bring some typical food products as gifts for your friends. What are three kinds of food products from the Netherlands that you are not allowed to bring into the US? [Government Regulation]
2. You are concerned with privacy issues related to electronic information and would like to know what laws have been passed by the US Congress regarding these issues. Identify three such laws. [Government Regulation]
3. A friend has a private well which is the family's only source of drinking water. Locate a US publication, which contains guidelines for the maintenance of safe water standards for private well use. [Health]
4. You are not sure about the safety of genetically engineered foods, and would like to find more information and research on this topic. Name four potential types of safety problems that have been raised. [Health or Project]
5. You are interested in learning more about what measures the US government has taken since 2001 to prevent Mad-Cow Disease. Identify three such measures. [Health or Project]
6. Name/find three research programs/projects that investigate the treatment/causes of dwarfism. [Project]
7. You are planning a cycling expedition along the Silk Road in Central Asia. Find a website that is a good source information about health precautions should you take. [Travel]
8. You are planning to travel to the northeast territories of India and wonder if there are any problems/restrictions for tourists. Find a website that is a good source of information about such problems/restrictions. [Travel]

The data collection on each searcher included:

- Pre-Experiment questionnaire - measuring gender, age, educational level, searching experience, and general computer usage
- Paper-Folding Test (VZ-1) - measuring spatial visualization trait
- Pre-searching answer and certainty
- Post-searching with answer and certainty
- Exit questionnaire - measuring understanding of and satisfaction with experimental process
- Questionnaire for User Interface Satisfaction (QUIS) - validated questionnaire of satisfaction with a computer user interface [6]

Successful completion of the task was determined by evaluating the user answer. Since some questions required more than one answer (e.g., food products from the Netherlands), each user's search was assigned a score, with two points for a complete answer, one point for a partial answer, and zero points for wrong answer. The grading of results was done by an OHSU graduate student.

Since we were focused on user factors, our analysis was carried out on the level of searcher, not individual questions. A correlation matrix for the four major measurements (VZ-1 score, pre-searching score, post-searching score, and QUIS score) was built using Pearson's correlation coefficient with two-tailed testing for statistical significance.

Results

We recruited the minimum 16 searchers from students in the computer science program at Portland State University and the medical informatics program at OHSU. Experiments were carried out at a computer laboratory at OHSU using PCs running Microsoft Windows 2000 and the Internet Explorer version 5.5 Web browser, connected to the campus computer network, which was in turn connected to the Internet.

The general characteristics of the searchers are shown in Table 2. This group was highly experienced in computer use and Web searching, although they had lesser experience searching on-line public access catalogs and bibliographic indexes. They were highly experienced with searching related to their work, but less experienced searching in the searching tasks for this study, such as health, shopping, and government policy. The group unanimously reported Google as their preferred search engine.

Table 2 - General characteristics of searchers.

Characteristic	**Average result**
Gender	9 male, 7 female
Age	median 18-27
Experience using computers (1-none to 4-some to 7-great deal)	6.8
Experience using Web (1-none to 4-some to 7-great deal)	6.5
Frequency of use for work tasks (1-never to 4-monthly to 7-daily)	6.8
Frequency of use for academic tasks (1-never to 4-monthly to 7-daily)	6.7
Frequency of use for personal tasks (1-never to 4-monthly to 7-daily)	6.7
Level of expertise with computers (1-novice to 7-expert)	6.0
Experience with Web search engines (1-none to 4-some to 7-great deal)	6.3
Experience with OPACs (1-none to 4-some to 7-great deal)	4.8
Experience with indexes (1-none to 4-some to 7-great deal)	2.8
Usually find what looking for when searching Web (1-rarely to 4-sometimes to 7-often)	6.3
Frequency of searching for work (1-never to 4-monthly to 7-daily)	6.3
Frequency of searching for shopping (1-never to 4-monthly to 7-daily)	4.6
Frequency of searching for traveling (1-never to 4-monthly to 7-daily)	4.4
Frequency of searching for medical/health (1-never to 4-monthly to 7-daily)	4.0
Frequency of searching for government policy (1-never to 4-monthly to 7-daily)	2.7
Frequency of searching for entertainment (1-never to 4-monthly to 7-daily)	4.8
Overall expertise with searching (1-novice to 7-expert)	5.7
Years searching	5.2 years
Favorite search engine	All 16 - Google

Table 3 shows the results of the major searcher-related variables. Table 4 shows the correlation matrix for those variables. with Figures 1-3 showing VZ-1, pre-searching score, and QUIS score plotted versus post-searching score. The largest correlation was 0.405 for pre-searching and post-searching scores, with an associated p-value of 0.12. Thus, no variable would enter a regression model and therefore additional analyses were not warranted. In order for a correlation coefficient of 0.40 to be significant with 80% power and a two-sided 5% significance level, a sample size of 47 would be required. The sample size was thus too small for any meaningful interpretation. The data were similarly analyzed using a non-parametric approach and comparable results were obtained.

Table 3 - Searcher-level analysis of major characteristics measured.

Searcher	VZ-1 Score	Pre-searching Score	Post-searching Score	QUIS Score
1	17.8	4	11	5.0
2	5.8	0	12	4.0
3	15	4	11	7.0
4	10	0	4	5.4
5	13	0	10	6.7
6	13.8	0	11	6.0
7	14	0	10	4.4
8	10	2	10	8.0
9	8.5	0	10	7.3
10	4.3	0	8	6.2
11	14.8	1	9	4.1
12	8.5	5	11	4.7
13	15	0	7	5.4
14	11	1	8	6.6
15	7	1	6	5.3
16	12.8	0	7	7.4
Average	11.3	1.1	9.1	5.9

Table 4 - Correlation matrix for searcher-level results.

	VZ-1 Score	Pre-searching Score	Post-searching Score	QUIS Score
VZ-1 Score	1	.23 .39	.16 .56	-.03 .91
Pre-searching Score		1	.41 .12	-.05 .87
Post-searching Score			1	-.09 .75
QUIS Score				1

Figure 1 - Scatter plot of VZ-1 score versus post-searching score.

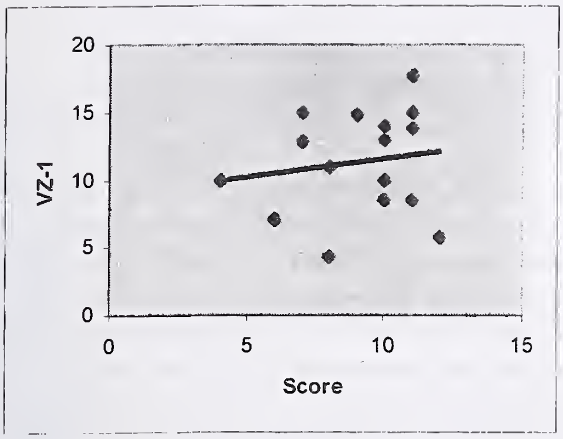


Figure 2 - Scatter plot of pre-searching score versus post-searching score.

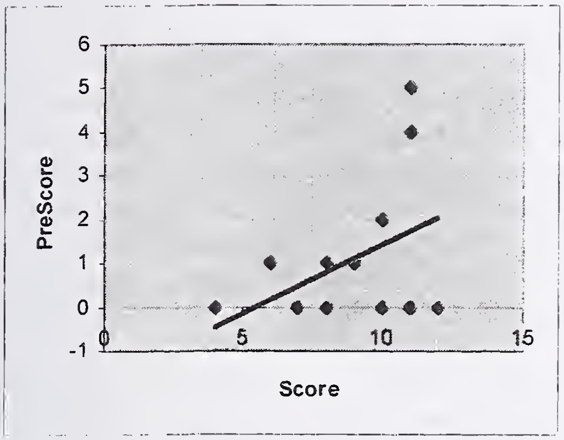
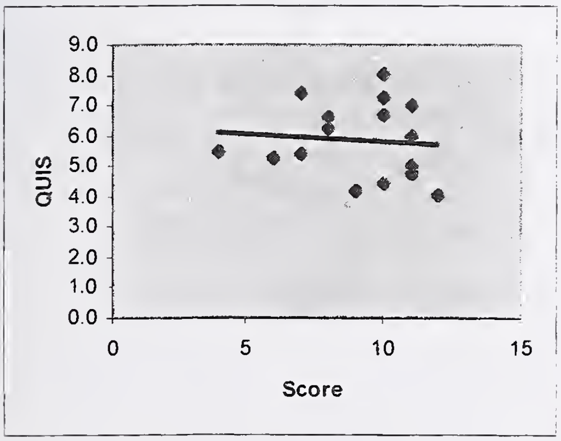


Figure 3 - Scatter plot of QUIS score versus post-searching score.



Conclusions

The OHSU results for the TREC 2002 Interactive Track showed some possible correlation between various user measures that did not reach statistical significance. Whether these measures were truly important could only have been assessed with a much larger sample size.

Statistical power is actually an overlooked challenge to evaluation of information retrieval systems. Even those carrying out batch-style evaluations must be concerned about it. Tague-Sutcliffe analyzed the results of the TREC-3 ad hoc experiments and found that the top half of the runs ranked by mean average precision had statistically insignificant differences with each other [7]. While Voorhees [8] has reassuringly found that results tend to keep their order even when different relevance judgments are substituted, Zobel has determined total recall is likely overestimated (i.e., additional relevant documents are likely to be found) [9]. In a related vein, Buckley and Voorhees analyzed the "stability" of results in batch studies and found a minimum of 25-50 queries needed to achieve it [10].

While most researchers who carry out evaluations in any field are probably familiar with statistical significance, which measures *alpha* error. and its meaning. Fewer research papers, however, report statistical power, which measures the minimization of *beta* error. Adequate statistical power is important in research, as an intervention in an experimental study may be of benefit, but the sample size is too small to tell. In fields such as medicine, the performance of "underpowered" clinical trials has been criticized, yet in many other experimental endeavors, researchers stop at reporting that results are "not statistically significant" [11]. Underpowered studies are a concern in TREC, especially given the nature of the venue, i.e., experiments performed on an annual cycle by research groups that do not generally have resources to carry out large-scale studies.

Another statistical challenge often overlooked in user-oriented IR evaluation studies is the non-independence of results from individual questions or topics. That is, analyses carried out at the level of individual question must take into account the fact that such questions are not completely independent, in that users search on multiple questions. In our previous experiments, we had to employ more complex statistical analyses to when evaluating factors at the level of the individual question [4, 5].

The results of our TREC 2002 Interactive Track experiments demonstrate that many measurable factors do influence the outcome of searching, but that sample sizes must be large enough to assess them well. The nature of the TREC experiments, with its short cycle for experimentation, can be at odds with adequately powered experiments. We hope to continue analyzing searchers once the .GOV collection has become stabilized.

References

1. Egan DE and Gomez LM, *Assaying, isloating, and accomodating individual differences in learning a complex skill*, in *Individual Differences in Cognition, Vol. 2*, Dillon R, Editor. 1985, Academic Press: New York.
2. Vincente KJ, Leske JS, and Williges RC, *Assaying and isolating individual differences in searching a hierarchical file system.* Human Factors, 1987. 29: 349-359.
3. Allen BL. *Cognitive differences in end-user searching of a CD-ROM index. Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1992. Copenhagen, Denmark: ACM Press. 298-309.
4. Hersh WR, et al., *Factors associated with success for searching MEDLINE and applying evidence to answer clinical questions.* Journal of the American Medical Informatics Association, 2002. 9: 283-293.
5. Hersh W, et al., *Challenging conventional assumptions of automated information retrieval with real users: Boolean searching and batch retrieval evaluations.* Information Processing and Management, 2001. 37: 383-402.
6. Chin JP, Diehl VA, and Norman KL. *Development of an instrument measuring user satisfaction of the human-computer interface. Proceedings of CHI '88 - Human Factors in Computing Systems.* 1988. New York: ACM Press. 213-218.
7. Tague-Sutcliffe J and Blustein J. *A statistical analysis of the TREC-3 data. Overview of the Third Text REtrieval Conference (TREC-3).* 1994. Gaithersburg, MD: National Institute of Standards and Technology. 385-398.
8. Voorhees EM. *Variations in relevance judgments and the measurement of retrieval effectiveness. Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1998. Melbourne, Australia: ACM Press. 315-323.
9. Zobel J. *How reliable are the results of large-scale information retrieval experiments? Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1998. Melbourne, Australia: ACM Press. 307-314.
10. Buckley C and Voorhees E. *Evaluating evaluation measure stability. Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* 2000. Athens, Greece: ACM Press. 33-40.
11. Halpern SD, Karlawish JHT, and Berlin JA, *The continuing unethical conduct of underpowered clinical trials.* Journal of the American Medical Association, 2002. 288: 358-362.

Question Answering Approach Using a WordNet-based Answer Type Taxonomy

Seung-Hoon Na, In-Su Kang, Sang-Yool Lee, Jong-Hyeok Lee

Department of Computer Science and Engineering, Electrical and Computer Engineering Division
Pohang University of Science and Technology (POSTECH)
and
Advanced Information Technology Research Center (AITrc)

1. Introduction

In question answering (QA), answer types are semantic categories that questions require. An answer type taxonomy (ATT) is a collection of these answer types. ATT may heavily affect the performance of QA systems, because its broadness and granularity provides coverage and specificity of answer types. Cardie [1] used 13 categories for entity classification, and obtained large performance improvement, compared with the method using no categories. Also, according to Pasca et al. [3], the more categories a system uses, the better performance the system shows. For example, consider two answer type taxonomies, A={PERSON}, and B={PRESIDENT, ENGINEER, SINGER, PERSON}. Given a question "*Who was the president of Vichy France?*", we know that the more specific answer type of this question is not PERSON, but PRESIDENT. Thus, if we use ATT B, a set of candidate answers from documents can be reduced to a set of PRESIDENT entities, by excluding the other PERSON entities such as ENGINEER and SINGER. This is not the case with ATT A. However, since ATT A cannot distinguish hypernyms of PERSON, the QA system should consider much more candidate answers.

Thus far, most QA systems rely on small-scale ATTs, with the number of semantic categories ranging from 20 to 100. Normally, these ATTs are created from a beginning set of frequently-asked answer types like person, organization, location, number, etc., and then they are incrementally extended to include unexpected answer types from new questions. However, these ad-hoc ATTs may raise the following problems in QA. First, it is nontrivial to manually enlarge a small ATT to a large one, as new answer types appear. Second, ad-hoc ATTs do not allow easy adaptation for processing questions asking new answer types. For such questions, the system needs to modify an existing IE module to classify entities into new answer types. Third, previous ATTs do not have sufficient broadness and granularity, where they are expected characteristics of ATT for open-domain QA.

Therefore, at this year's TREC, we have taken a question answering approach that uses WordNet itself as ATT. In other words, our QA system maps an answer type into a concept node called a synset in WordNet. WordNet provides sufficient diversity and density to distinguish specific answer types for most questions. By using such an ontological taxonomy, we do not have the above problems with small ad-hoc ATTs.

This paper is organized as follows. Section 2 describes each module of our QA system, and section 3 shows TREC-11 evaluation results, and concluding remarks are given in section 4.

2. System Description

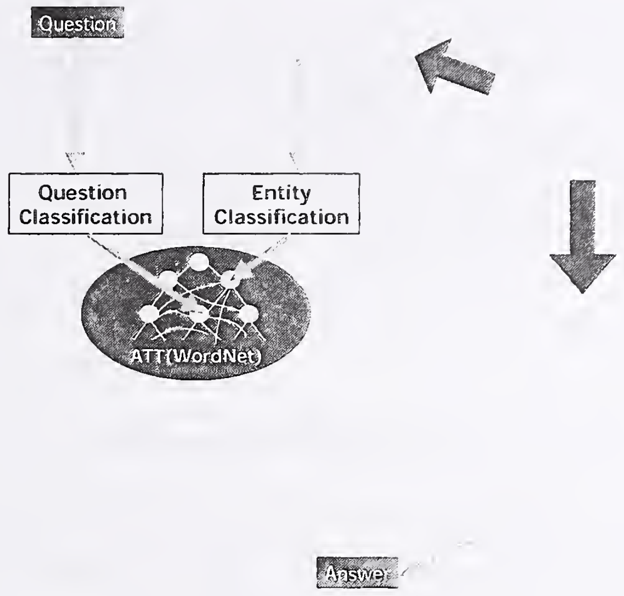


Figure 1. System Architecture

Our QA system consists of components as illustrated in figure 1. *Question Classification* analyzes a question to determine its answer type which maps to a synset in WordNet. *Passage Retrieval* formulates a vector query from the question, and retrieves relevant passages. *Entity Classification* processes top N passages from *Passage Retrieval* to classify each entity into its semantic category which as well corresponds to a synset in WordNet. A final answer is obtained from Answer Extraction by processing two sub sequent steps: *Determination of Candidate Answers* and *Ranking Candidate Answers*. *Determination of Candidate Answers* semantically recognizes all candidate answers in relevant passages by matching a taxonomic relation between the answer type of a question and the entity type of each entity, and regard matched entities as candidate answers. In the matching process, if any entity type is a hypernym of the answer type, since the entity does not have sufficient evidence to be selected as a candidate answer, thus in the case, *Entity Feedback* is invoked to collect clue words indicating more specific type from whole document collection. After *Entity Feedback*, the entity can be classified to more specific semantic category by *Entity Classification*. Unfortunately, in TREC-11, we did not perform experiment on the *Entity Feedback*. *Ranking of Candidate Answers* ranks candidate answers by calculating the similarity between a question and the passage containing each candidate answer, and generates a top candidate answer.

2.1 Question Classification

For each question, *Question Classification* determines its answer type using a set of 20 question pattern rules that were created from analyses of previous TREC questions. First, a question is tagged with part-of-speech tags using Tree-Tagger [6]. Next, our NP-chunker performs NP chunking and marks a linguistic head of a noun phrase. The NP-chunker was designed by a simple rule-based scheme. After chunking, question pattern rules are applied to the question to determine an answer type of a question.

Lexical constraint	Semantic constraint	Answer type	Example
What (be) NP	{head of NP} <= $\text{ENTITY}_{\text{synset}}$	{head of NP}	*What is the president of U.S. ?*
How many NP	{head of NP} < $\text{UNIT}_{\text{synset}}$	{head of NP}	*How many miles ...?*
How JJ	$\exists Y$ *Value_of* ({JJ}, X) and *Unit_of* (X,Y)	*Y*	*How fast does a cheetah run?*
who		$\text{PERSON}_{\text{synset}}$	*Who was the lead singer for the Commodores?*

Table 1. Question Pattern Rules

Table 1 shows some examples of question pattern rules, where lexical and semantic constraints describe preconditions to be met before the question pattern is matched against a question. $\{X\}$ indicates a WordNet synset of a lexical word corresponding to X. Since there are several senses for a lexical word, $\{X\}$ is currently assumed to be the most frequent sense. '$\{X\} < \{Y\}$' means that $\{X\}$ be a hyponym of $\{Y\}$. For example, suppose a question "*What is the governor of Colorado?*". After performing np-chunking for this question, the question is converted into "*What is [the governor]$_{NP}$ of [Colorado]$_{NP}$?*". This is matched with the first rule in table 1, because it satisfies a lexical constraint '*What (be) NP*', and a synset of the *NP* head $\text{GOVERNOR}_{\text{synset}}$, is a hyponym of $\text{ENTITY}_{\text{synset}}$. In this example, therefore, the answer type is a synset $\text{GOVERNOR}_{\text{synset}}$ in WordNet.

On the other hand, to deal with questions starting with 'how', we defined an additional semantic relation *Unit_of* into WordNet as follows.

DISTANCE$_{synset}$ — *Unit_of* → *LINEAR_UNIT*$_{synset}$

FINANTIAL_CONDITION$_{synset}$ — *Unit_of* → *MONETARY_UNIT*$_{synset}$

This relation is also used in question pattern rules. For example, given a question "*How far would you run if you participate in a marathon?*", it satisfies the lexical constraint of the third pattern in table 1, where *JJ* is a part-of-speech tag for an adjective. Because a synset $\text{FAR}_{\text{synset}}$ for 'far' has a *Value_of* relation with a synset $\text{DISTANCE}_{\text{synset}}$, the variable *X* in the semantic constraint becomes $\text{DISTANCE}_{\text{synset}}$. Since, we defined that $\text{DISTANCE}_{\text{synset}}$ has a *Unit_of* relation with $\text{LINEAR\_UNIT}_{\text{synset}}$ in WordNet, the variable *Y* becomes $\text{LINEAR\_UNIT}_{\text{synset}}$. Therefore, the answer type of the question is set to $\text{LINEAR\_UNIT}_{\text{synset}}$.

2.2 Passage Retrieval

Passage Retrieval consists of two processing steps: document retrieval and passage ranking. For document retrieval, we have developed an IR system based on a vector space model. To extract terms, we use a stop word list provided by the SMART system [5], from which 30 words such as *first*, *second* are eliminated because they provide important clues in QA. We do not use stemming, but use Tree-Tagger [6] to obtain root forms which are used as terms. To weight each term, we employ the weighting formula *nxx.bfx* introduced by Salton and Buckley [4].

From top 1000 documents generated by a document retrieval module, passage ranking identifies all legitimate passages in each document and rank them, using cover density ranking by Clarke et al. [2]. A passage unit is the minimal number of subsequent sentence including maximal number of query terms. No three passages are taken from the same document and no passages can contain more than five sentences.

2.3 Entity Classification

From the top 15 passages obtained by passage retrieval, *Entity Classification* classifies each entity occurring in the passages into a semantic category that maps to a synset in WordNet. First, we access entity databases. There are three kinds of entity databases: person names, location names, and organization names. If an entity is found in one of these databases, then the type of the entity is set to the type of the entity database. Otherwise, the entity is looked up in WordNet. and its corresponding synset is selected as the entity type. If the entity is not found even in WordNet, a clue word related to the entity is searched within the document containing the entity. Here, a clue word means a word that can indicate a semantic category of the entity. To recognize a clue word, we use a set of 30 type indicator patterns, which was empirically constructed. Table 2 shows some examples of type indicators. For instance, appositive constructions may provide type words for entities. For entities like "*Nancy Powers*" and "*Thomas*" in table 2, we know that their entity types are $\text{DIRECTOR}_{\text{synset}}$ and $\text{DETECTIVE}_{\text{synset}}$ respectively, since the entity and its type word are grouped into an appositive construction.

Type indicator	Examples
Appositive	1) **Nancy Powers**, a sales director for a medical company 2) **Steve Thomas**, the public detective
Role	1) Mrs. **Aquino**, 2) **New York** city
Relative clause	1) **Kenneth Starr**. who
Preposition	1) in **Washington**
Unit	1) **8,000** miles, 2) **8** miles per hour

Table 2. Examples of Type Indicators

2.4 Determination of Candidate Answers

Determining whether an entity is a candidate answer or not is based on taxonomic relations in WordNet between an entity type and the answer type of a question. Figure 2 shows three possible taxonomic relationships between the entity type and the answer type.

t_q : Answer type t_e : Entity type

A > B : A is a hypernym of B

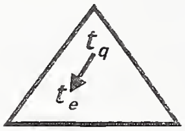
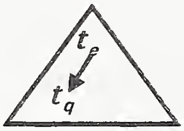
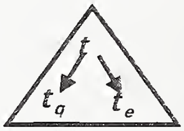
Taxonomic Relationship	1) $t_q \geq t_e$	2) $t_q < t_e$	3) $t > t_e, t > t_q$
			
Candidate Answer?	Completely Candidate Answer	Ambiguous: Need more local contexts	Not Candidate Answer

Figure 2. Taxonomic Relationships between an Answer Type and an Entity Type

The first is the case where the answer type is a hypernym of an entity type. The second is the case where an entity type is a hypernym of the answer type. The third includes all the other cases. As an example of the first case, suppose that the answer type is PERSON$_{synset}$ and an entity type is ENGINEER$_{synset}$. Then, since PERSON$_{synset}$ is a hypernym of ENGINEER$_{synset}$, the entity becomes a candidate answer. As an example of the third case, suppose that the answer type is ENGINEER$_{synset}$ and an entity type is PRESIDENT$_{synset}$. In this example, because the entity is not an ENGINEER$_{synset}$, it cannot become a candidate answer. As an example of the second case, if the answer type is PRESIDENT$_{synset}$ and an entity type is PERSON$_{synset}$, it is not clear whether the given entity is a PRESIDENT$_{synset}$ or not. So. candidate answer determination fails. In this case, an entity feedback module is fired to acquire another clue words from the whole document collection, which may contain the more specific semantic category for the same entity.

2.5 Ranking of Candidate Answers

After recognizing candidate answers. each candidate answer is ranked by the formula (1).

$$Score = Type\_score \times Context\_score$$

$$Context\_score = \sum_{(t_i,t_j)\in T} w_{q(t_i,t_j)} \times w_{d(t_i,t_j)} \quad \text{--- (1)}$$

$$T = Second\ order\ term\ set$$

In formula (1), *Type_score* is determined by WordNet taxonomic relationships between the answer type of the question and the entity type of the candidate answer. When the relationship belongs to the first case in figure 2. its *Type_score* is 1, and in the second case, *Type_score* set to 1/2 since currently our QA system does not support entity feedback module.

Context_score is a score calculated from proximity between the candidate answer and the question words in the passage where the candidate answer appears. To compute a context score, first we convert the passage into a second-order vector (SOV) that is different from the traditional first-order vector (FOV). The difference is that a term in FOV consists of a single lexical word and a term in SOV consists of two lexical words. For example, when a vocabulary set is V ={A, B, C}, the possible term set in SOV is T = {AB, AC, BC}, where each element is called by a second-order term. To weight a second-order term, we assume a proximity hypothesis that the closer two participating lexical words of the second-order term is in a passage, the more important the term is. According to this hypothesis, given any two lexical words t_i and t_j in second order term set, we calculate the weight of the second-order term (t_i,, t_j) by formula (2), where $sent\_dist_{d,(t_i,t_j)}$ and $pos\_dist_{d,(t_i,t_j)}$ is a sentence distance and a positional distance between t_i and t_j on the passage d respectively.

$$w_{d,(t_i,t_j)} = proximity_{d,(t_i,t_j)} \quad \text{--} \text{(2)}$$

$$proxmity_{d,(t_i,t_j)} = \begin{cases} \dfrac{1}{sent\_dist_{d,(t_i,t_j)} \times \log\,(pos\_dist_{d,(t_i,t_j)}+1)} & \text{if } t_i \text{ and } t_j \text{ exists in the passage } d \\ 0 & otherwise \end{cases}$$

For the question q, we use formula (3). where $pos\_dist_{q,(t_i,t_j)}$ is a positional distance between t_i and t_j on the question q.

$$w_{q,(t_i,t_j)} = \left(\frac{idf_{t_i} + idf_{t_j}}{2}\right) \times proximity_{q,(t_i,t_j)} \quad \text{--} \text{(3)}$$

$$proximity_{q,(t_i,t_j)} = \begin{cases} 1 & \text{if } t_i \text{ and } t_j \text{ exists in the same noun phrase within the question } q \\ \dfrac{1}{pos\_dist_{q,(t_i,t_j)}} & \text{if } pos\_dist_{q,(t_i,t_j)} < 3 \\ 0 & otherwise \end{cases}$$

NIL answers are generated as follows. If either a proper noun in a question does not appear in the top passages, or the confidence value for the top answer is below a threshold 0.1, then our system generates the NIL answer. The confidence value for NIL generation is calculated by formula (4). Finally we rank all the top answers with their confidence values.

$$Confidence\_value = \frac{\sum_{(t_i,t_j)\in T} w_{q,(t_i,t_j)} \times w_{d,(t_i,t_j)}}{\sum_{(t_i,t_j)\in T} w_{q,(t_i,t_j)}} \quad \text{--} \text{(4)}$$

3. Evaluation Results

We obtained the following evaluation results at TREC-11.

Number of wrong answers	399
Number of unsupported answers	5
Number inexact answers	10
Number right answers	86
Confidence-weighted score	0.298
Precision of recognizing no answer	0.161 (=15 / 93)
Recall of recognizing no answer	0.326 (=15 / 46)

We selected randomly 100 of 500 questions, and evaluated the performances for some well-known answer types. The results are shown in table 3. Here, an answer type includes all subtypes of the answer type. We had a promising performance in the case of an answer type $DATE_{synset}$, but we did not generate correct answers in the case of other entity types such as $HORSE_{synset}$, CAR_{synset}.

Answer type	Percentage	Precision
person	19%	21.05%
location	23%	8.69%
organization	4%	25%
unit	7%	14.28%
count	4%	0%
date	17%	58.82%
fullname	1%	0%
other entities	25%	0%
Total	100%	18%

Table 3. Performance by Answer Types

4. Conclusion

Our QA system used WordNet as ATT for open-domain question answering. Question pattern rules were employed to determine an answer type of a question. In addition, in order to map entities in relevant passages into an entity type that corresponds to a synset in WordNet, we devised type indicator patterns. For each entity, taxonomic relationships between the answer type and its entity type are checked to qualify candidate answers. Unqualified entities are passed to an entity feedback module which provides several clue words to determine the more specific type for the problematic entity. A final answer is obtained from a ranking method, which is based on a second-order vector representation for relevant passages.

In our unpublished experiments on 300 questions in TREC-8,9,10, our system showed about 10% performance improvement by using entity feedback. But on TREC-11 questions, we did not yet perform experiments. From TREC-11 results, we had recorded a bad performance for non-basic entities like $\text{CAR}_{\text{synset}}$, $\text{WEAPON}_{\text{synset}}$. In future, we plan to refine the entity classification techniques to determine types of these non-basic entities effectively, and to apply the entity feedback method on such a classification method.

Acknowledgements

This work was supported by the Korea Science and Engineering Foundation (KOSEF) through the Advanced Information Technology Research Center (AITrc).

References

[1] Cardie, C., Ng, V., Pierce, D., and Buckley C., "Examining the Role of Statistical and Linguistic Knowledge Sources in a General-Knowledge Question-Answering System", In Proceedings of the 6th Applied Natural Language Processing Conference, pp.180-187, 2000

[2] Clarke, C.L.A., Cormack, G.V., and Tudhope, E.A., "Relevance Ranking for One to Three Term Queries", Information Processing and Management, 36(2):291-311, 2000

[3] Pasca, M.A., and Harabagiu, S.M., "High Performance Question / Answering", In Proceedings of the 24rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.366-374, 2001

[4] Salton, G., and Buckley, C., "Term-weighting Approaches in Automatic Text Retrieval", Information Processing and Management, 24(5):513-523, 1988

[5] SMART system: ftp://ftp.cs.cornell.edu/pub/smart/

[6] Tree Tagger: http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/

TREC2002 Web, Novelty and Filtering Track Experiments using PIRCS

K.L. Kwok, P. Deng, N. Dinstl & M. Chan
Computer Science Department, Queens College, CUNY
Flushing, NY 11367

1 Introduction

In TREC2002, we participated in three tracks: web, novelty and adaptive filtering. The Web track has two tasks: distillation and named-page retrieval. Distillation is a new utility concept for ranking documents, and needs new design on the output document ranked list after an ad-hoc retrieval from the web (.gov) collection. Novelty track is a new task that involves identifying relevant sentences to a question, and to remove duplicate or non-novel entries in the answer list. The third track is adaptive filtering. We revived a filtering program that was functional at TREC-9 with some added capability. Sections 2, 3, 4 describe our participation in these tracks respectively. Section 5 has our conclusion.

2 The Web Track

This year the web track involves two tasks: topic distillation and named-page finding. Named-page finding is similar to last year's home page finding [1] except that an answer page may be a sub-site address containing what the user wants that is named in the query. Topic distillation is new, and is concerned with locating the most useful pages (out of many) that best and comprehensively describe a user's topic, either by content or via links. Previous investigations on topic distillation such as [2,3,4,5] mostly tie the process to 'quality' identification. They employed Kleinberg's HITS [6] algorithm as the primary method, and added content weighting as secondary improvement. Authority and hub pages found were identified with topic distillation answers. In this experiment, we employ page content weighting (including anchor texts) as our primary process, and add out-link content weight to help determine answers. This is based on the description of the task as given in the Guidelines for TREC-2002 Web Track (http://trec.nist.gov).

The collection for this years' web task is the .gov collection, a recent crawl (early 2002) on the government domain web pages. It consists of nearly 1.3 million pages totaling about 10 GB. The file was processed to our internal format and broken up into about 3 million sub-documents. A dictionary of over 5.8 million terms was produced including some 2-word phrases. This was truncated to about 1.4 million by ignoring terms with frequency 2 or less, or greater than 600,000. As usual, 50 topics (later truncated to 49) were used for retrieval. We experimented with short queries employing only the title section of each topic as queries. They averaged to ~3.5 terms after stemming and removal of stop-words.

2.1 Improved Web Retrieval

Over the last two TREC conferences, PIRCS has provided about average performance in the Web track. The Web10g collection scale is much larger than the 2 GB that we have been accustomed to in previous ad hoc tracks, and the web page genre is very different from newspaper type. We spent some effort to try to analyze the situation, and test various parameter settings in order to understand the problem and to improve retrieval results for 10-GB scale web collection. It turns out that the major cause of lackluster web retrieval performance with PIRCS is due to a wrong setting of the high Zipf threshold that is used for screening out high frequency indexing terms – so called statistical stop-words. This threshold was previously set at 180,000 (about 18% of the number of documents) in order to gain better efficiency with our network implementation of PIRCS. After upgrading our system with 512 MB of memory and setting this threshold at a high 500,000 to include more terms, mean average precision (MAP) improved substantially for both short and long queries for Trec-9 and Trec-2001 web experiments as tabulated in Table 2.1 due to this single parameter change. Loss of indexing terms is a major cause for unsatisfactory results. Additional gains ·were observed, when pseudo-relevance feedback parameters were optimized, for example. The improved procedures are employed for this year's web tasks.

Web track Trec-	Short (title)	Long (all sections)
2001 (old Zipf threshold	0.1742	0.1715
2001 (new Zipf threshold)	0.2039	0.2054
9 (old Zipf threshold)	0.1750	0.2209
9 (new Zipf threshold)	0.1818	0.2448

Table 2.1 Improved Web Retrieval Results

2.2 Distillation Task

According to the track description, the purpose of topic distillation is to find the 'key resource' page(s) for a given topic. The concept of 'key resource' has been described in the Guideline for TREC-2002 Web Track (http://trec.nist.gov). Examples may be a page with outstanding content. or one with out-links to good content pages on the topic. Content may be less important than useful links in a page, and in general answers are diversified so that a relevant host site may not have many distillation page(s).

Our strategy for this task is to a) first find the best content pages for a topic; and b) add link processing to find diversified key resources among these pages. The first step makes use of our normal ad-hoc retrieval ranking since it is content-oriented. The second step involves identifying the importance of linked content for each page. These steps are described below.

To make use of the structured property of web data, we create four different collections by separating each web page into four objects identified by the same DocID: title, text, meta and href objects. 'Title', 'text', and 'meta' (whose metadata content is usually not for display) are obtained from the appropriate tag fields of the page. For the 'href' collection, each document is composed of anchor texts from different pages that link to one particular URL. This URL is then mapped to a unique DocID using the 'url2id' file provided. 'href' therefore defines a page based on its in-link anchor content irrespective of what the page itself may contain. The 'text' and 'href' collections are processed with Porter's stemming, while 'title' and 'meta' are left un-stemmed. The purpose is to obtain higher precision with the latter two shorter documents.

We form a query from only the title field of a topic. This is a required submission. The query (stemmed or un-stemmed) is used to rank items from each of the four collections using our PIRCS system, and four ad-hoc retrieval lists are obtained.

To satisfy the desired diversified key resource property, we form host groups. A host group contains pages having the same host address. The DocID of each retrieved page is converted to URL. URL addresses allow us to merge and categorize pages into a set H of host groups each with varying number of pages. Since relevant content documents usually occur in the top part of a retrieval list, we limit key resource finding to the top 100 pages of each of the 4 lists except for 'meta', which is limited to the top 10. The 'meta' collection may be less reliable than the others. These form a best-page candidate pool and organized into host groups.

Each unique page has four normalized retrieval status value values (RSV) (including zero when it does not appear on some retrieval lists). Each RSV is normalized to lie between 0 and 1 by dividing by the sum of the top 1000 RSV's. Later, another normalization based on transformation by the function g(RSV) = exp(a+b*RSV)/[1+exp(a+b*RSV)] was tried and it performs better. We combine the normalized RSV values to form a weight called **A-wt (content)** for a page according to the following criteria:

```
If (page-type== graphic ('giff', etc.))
    A-wt=0

else  if (page-type==HTML)
    A-wt = 0.4*title.RSV + 0.4 href.RSV
    + 0.15* text.RSV + 0.05*meta.RSV

else  if (page-type==non-HTML(pdf, etc))
    A-wt = 0.2*text.RSV + 0.8*href.RSV
                                        (1)
```

We assume that the A-wt can characterize roughly how content-relevant a page is to the retrieval topic. Another weight called **B-wt (link)** is also assigned to each page based on its out-links and defined as follows:

$$\text{B-wt} = \Sigma_{\text{out-links}} (\text{A-wt}) \qquad (2)$$

The sum is over links pointing within the candidate pool only, not to the collection. We assume the B-wt can characterize roughly how strong a page's link content is and its contribution to its distillation power.

Each member h of H is also assigned a weight equal to the $\Sigma_{\text{pages-in-h}}$ (A-wt)/sqrt(n) for all the n pages within the host. Thus, host groups can be ranked for content. Within each group, pages are ranked by their combined (A-wt + B-wt), which we call **page weight**. Thus, a page may have little content (i.e. small A-wt), but if it points to many useful pages, its B-wt can be large, ranking it higher among peer pages within a group based on its page weight. A picture of the candidate pool organized as weighted host groups is shown in Fig.2.1.

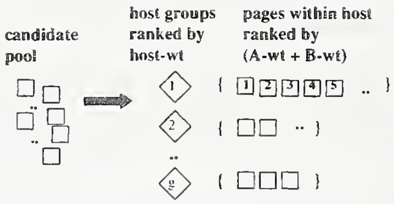

candidate pool
host groups ranked by host-wt
pages within host ranked by (A-wt + B-wt)
1
2
g
{ 1 2 3 4 5 .. }
{ .. }
..

Fig.2.1 Weighted Pages within Weighted Host Groups

To form the answer list for the distillation task, we adopt two strategies resulting in our submissions pirc2Wd1 and pirc2Wd2 (the tag has the meaning: pircs-year02-web-distillation-run#). For pirc2Wd1, the top one page from each of the best 10 host groups are listed first, followed by padding it with other pages sorted by page weights. The second submission pirc2Wd2 uses the top 2 pages from each of the top 10 host groups, sorted by page weight to define the top 20 answers, then followed by padding as in pirc2Wd1.

Our approach of forming weighted host groups and then using page weight to sort pages within a group, is designed to find key resource page(s) within the most contextually relevant hosts. Forming the answer list by selecting top page(s) from each group is designed to allow diversification in our distillation answer list. Other methods to form answer lists may also be employed.

Results and Discussions

Table 2.2 presents official evaluation of our distillation experiments using precision at 10, 20 and 30 documents retrieved values. It is seen that the second approach of selecting 2 top pages from each host group has much better performance, especially at P10 (0.1082 vs. 0.0816). The first approach suffers from too much diversification because: a) it is risky to assume that statistical ranking can always position key resource pages to the top of each group; b) many queries do have multiple same-host answers, and output of single page from each host artificially diminishes the chance of putting key resources in top 10.

	P10	P20	P30
pirc2Wd1	.0816	.0765	.0633
pirc2Wd2	.1082	.0857	.0741

Table 2.2: Web Distillation Results (Submitted)

After results are known, we fix a bug in our program and employ better RSV normalization to attain a P10 value of 0.1204 as shown in Table 2.3. We had thought that 'title' and 'href' retrieval would be more accurate and weigh them higher in (1). In reality, 'text' retrieval remains far superior. When the coefficients for combining RSV's among the four collections to define A-wt were set to 0.65 ('text'), 0.15 ('href'), 0.15 ('title'), and 0.05 ('meta'), and also normalizing the B-wt by the number of out-link edges, the P10 value jumped to 0.1673. When 3 pages are selected from each host (instead of 2), or with no host restriction (just use A-wt + B-wt for ranking), P10 values continue to improve to 0.1735 and 0.2204 respectively. However, when only the content weight (A-wt) is used, also ignoring host groups, distillation result is very similar to using A-wt + B-wt. It seems that a) out-link content (B-wt) is not necessary for key resource detection (using A-wt only performs almost as well); and b) host grouping leads to worse performance. The latter point should be viewed in the context that out of 1574 key resource answers for the 49 topics, 432 have unique host, 112 have duplicate hosts, 55 have three, and the rest (~48%) share four or more same host. Restricting result list from diverse host groups would depress the chance of getting relevant answers within the top 10 retrieved.

	P10	P20	P30
bug fix, better RSV normalization	.1204	.1000	.1075
better combination coeffs., normaliz B-wt	.1673	.1296	.1381
3 top pages each host	.1735	.1551	.1367
No host: A-wt + B-wt	.2204	.1816	.1517
No host: A-wt	.2184	.1837	.1490

Table 2.3: Web Distillation Results (Post-Evaluation)

2.2 Named-Page Task

The objective of the named-page task is to retrieve an appropriate page(s) that contains answers to wanted item(s) named in a query. There are 150 topics and they all vary between two to six words long. We submitted two runs for this task based on the processing methodology of the distillation task called: pirc2Wnp1 and pirc2Wnp2. The first method outputs 50 top documents from the collections 'title' and 'href' (total 100). A-wt is defined for each page, and the top 50 according to A-wt is returned as the answer list. The second method selects top 10 from the 'meta' collection, top 100 from each of 'title', 'text' and 'href' collections (total 310). These are grouped into hosts as in distillation task. Top 5 pages from each of top ten hosts are selected; these are sorted by page weight and returned as the answer list.

	Pirc2Wnp1	Pirc2Wnp2
#of topics having answer ranked 1	30	3
2	5	4
3	6	3
4	3	5
5	6	4
6	1	4
7	2	2
8	1	2
9	6	2
10	1	2
MRR	0.263	0.077
#topics with ans. ≤rank 10	61(40.7%)	31(20.7%)
#topics with ans. ≤rank 50	95(63.3%)	65(43.3%)
#topics with ans. not found	55(36.7%)	85(56.7%)

Table 2.4: Web Named-Page Results (submitted)

Results and Discussions

Table 2.4 summarizes results of the two runs. Mean reciprocal rank (MRR) is the measure for evaluation. It is seen that method 1, pirc2Wnp1, has much better performance (MRR = 0.263) than the second method (MRR = 0.077). Just using the top 'title' and 'href' items from their retrieval lists returns a fair number of the answers (~41%) within top 10. Organization into host groups is not an appropriate strategy for this content-oriented task. Ranking by content (A-wt) is sufficient to bring about reasonable performance.

The lackluster result can be traced again to our wrong emphasis on the 'title' and 'href' collections only. After results are known, we change our processing to include 50 documents each from the collections except 'meta', use the modified combination coefficients to define A-wt as discussed in Section 2.2, and output the top 50 according to A-wt. The MRR value doubled to 0.525, and 96 queries had correct answers in top 5.

3 The Novelty Track

A new track called novelty task is defined this year. Given a query, its objective is to first rank and detect relevant sentences from a given set of sentences (that have been obtained from relevant documents of the query). The system next tries to identify among these sentences in an ordered fashion, those that contain novel information -- those not novel are removed from the list. This is done after sorting the relevant sentences by document and sentence# order. The objective of this task has similarity to previous work done such as duplicate document removal in IR [7], first-story detection in TDT [8], or redundancy detection in adaptive filtering [9].

For this experiment, we employ all sections of a topic to form long queries for retrieval because the 'documents' are actually short sentences. The queries average to 19.14 unique terms. Since the sentences come from relevant documents of TREC-8, we use the TREC-8 dictionary to provide better statistics for processing and retrieval. However, the high Zipf threshold has been reset to 400,000 to include more high frequency terms as discussed in Section 2.1.

Only initial retrieval without pseudo-relevance feedback was performed. Based on experimentation with the four training topics, we test two RSV threshold (tr) values on the ranked retrieval list to help decide on the relevance of retrieved sentences: submission pircs2N0{1,2} employ tr=1.25, and pircs2N0{3,4} use tr=1.5. Thus, retrieved sentences with RSV > tr are considered relevant.

This set of relevant sentences is sorted according to DocID and sentence#. For each sentence, every one of its un-stemmed words is expanded with synonyms by consulting with WordNet. All senses of the noun type are used. The resultant set of words is sorted, and duplicates removed. A double loop passes down the sentence list, and a novelty coefficient based on the Dice formula is evaluated for each pair of sentences Si and Sj:

$$v = \text{Novelty coeff.} = \frac{|\,\text{Si interset Sj}|}{|\text{Si union Sj}|} \qquad (3)$$

If v < a threshold tv, Sj is considered novel with respect to Si, otherwise Sj is removed. pircs2N01 and pircs2N03 employ a threshold tv=0.35 (originally documented as 0.3), and pircs2N02, pircs2N04 use tv=0.5. In addition, a fifth submitted run pircs2N05 does not use synonyms, just raw words, and acts as control with thresholds set to tr=1.5, tv=0.3.

Results and Discussions

Five runs were submitted to the novelty track labeled as pircs2N??, where ?? range from 01 to 05. Except for pircs2N05, all runs employ WordNet to find synonyms to words in the retrieved relevant sentences to decide for novelty. Results of the submitted experiments concerning decision on relevance is shown in

pircs2N	tr	P	R	ΣPq*Rq
{01, 02}	**1.25**	**.16**	**.49**	**.08**
{03, 04, 05}	1.5	.18	.4	.072

Table 3.1: Relevant Sentence Decision Results (submitted)

Table 3.1. The average precision (P) and recall (R) effectiveness are evaluated for relevant sentences at the two RSV threshold values tr. Official measure for this task is ΣPq*Rq, i.e. sum of the product of precision and recall for each query q. It is seen that a lenient value of tr = 1.25 returns 0.49 recall ratio but a low 0.16 for precision. However, their product ΣPq*Rq leads to 0.08, better than the 0.072 product when the tighter threshold tr = 1.5 was used.

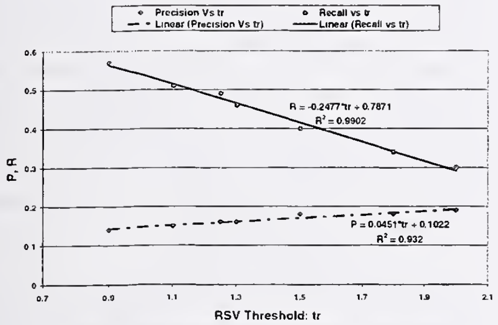


Fig.3.1: Variation of P, R vs Relevance Threshold tr

Fig.3.1 plots the variation of P and R vs. threshold tr. Although P*R is not the same as ΣPq*Rq, one can nevertheless gains some idea of the result. P and R have very good linear fit for this retrieval environment. It shows that as RSV threshold tr changes from 1.25 to 1.5, P*dR/d(tr) drops faster than R*dP/d(tr) rises, leading to a fall of P*R value. If tr were set to 1.1 (or less), ΣPq*Rq improves to a value of 0.81.

After relevance determination, the set of sentences is passed to novelty processing. Results of using two novelty thresholds: tv = 0.35 and 0.5 are shown in Table 3.2. Two corrections need to be pointed out: 1) book-keeping of the files during submission were mixed up and the tv threshold for runs pircs2N01 and 03 should have been 0.35 instead of 0.3 as documented during submission; 2) the submitted run pircs2N05 (which was supposed to be WordNet free) was actually identical to run 03. The correct run denoted as 05* was not submitted, but its result is shown in Table 3.2. From the table, it is seen that pircs2N02 has the better result among the 5 submissions. For our system, it seems preferable to increase the novelty threshold tv to 0.5 (rather than 0.35) so that two sets of sentence words (appropriately expanded with WordNet synonyms) need to have larger overlap before they are considered similar and not novel (3). This, together with an RSV threshold of tr=1.25 produces a ΣPq*Rq value of 0.069.

pircs2N	tr	tv	P	R	ΣPq*Rq
01	1.25	.35	.15	.39	.062
02	**1.25**	**.5**	**.15**	**.43**	**.069**
03	1.5	.35	.17	.31	.056
04	1.5	.5	.17	.36	.064
05==03	1.5	.35	.17	.31	.056
05*	1.5	.35	.17	.37	.066

Table 3.2: Novel Sentence Decision Result (submitted except for the corrected *05)

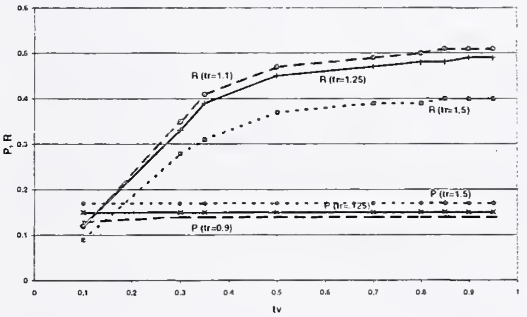


Fig. 3.2: Variation of P,R vs Novelty Threshold tv

We have plotted the variation of novelty precision and recall values against the threshold tv in Fig.3.2 for three relevance thresholds 1.1, 1.25 and 1.5. It is seen that novelty precision value P is practically constant over a large range of tv values, and they do not vary too much with respect to the tr threshold: 0.14 to 0.17. Apparently, as the tv threshold is changed. correctly identified novel sentences and incorrect ones are included at the same rate. However, as more sentences are accounted, novelty recall improves. This suggests one should set the relevance threshold tr low (like 1.1 or lower) to recall more relevant sentences, and also set the novelty threshold tv high (like 0.9) to include more sentences as novel. At tr=0.9 and tv=0.9, the official measure ΣPq*Rq evaluates to a value of 0.76, an improvement of nearly 12% over our best submitted results. This is achieved based on high recall values. Precision ratios are low at 0.14 to 0.17.

The last line in Table 3.2 (pircs2N05*) shows novelty detection of sentences without WordNet expansion of terms. The un-stemmed words were used for overlap calculation (3). It returns a value for ΣPq*Rq of 0.66, about 18% better than pircs2N03, showing that WordNet expansion is not good at these parameters. However, at the better parameters of (tr, tv)=(0.9, 0.9) they all return a ΣPq*Rq value of 0.76. If stemmed words were used, slightly worse performance was observed.

As an example of WordNet expansion, we illustrate (for Query 305 "most dangerous vehicles") with sentence #20 of document LA031689-0177: "stresses safe driving". These three words expand to: {emphasis, accent, tension, tenseness, stress, focus, strain}, {condom, rubber, safety, safe, prophylactic} and {drive, driving}. Thus good synonyms are brought in as well as many bad ones. A filter needs to be built to screen out unwanted senses.

4 Adaptive Filtering Track

This year's adaptive filtering task makes use of the topics numbered R101-R200 to select documents in date order from the Reuter collection for the period October 1, 1996 to July 31, 1997. Adaptive filtering is difficult. A possible approach is to use a two-step strategy. At start when little knowledge is known, a simple adaptive threshold adjustment and profile re-weighting method is used. Later when sufficient relevant data is available, expand and train the profile to increase the prospect of selecting only the relevant ones. We also employ the dictionary from last year's Q&A collection (in addition to those from training documents) as a basis for processing to ensure that most terms from the test collections are included.

Many considerations are needed for adaptive filtering. These include defining an initial profile together with an initial selection threshold to start the process, dynamically adapt the threshold to select or not to select a document for examination, adaptively train and expand the profile to tailor to the type of documents seen so far, determine how often these changes are to be made, and at the same time attempt to maximize the utility value. Apparently the adaptation of the filtering profile and that of the threshold are both useful. Improved profile does a better job in separating the relevant documents from the irrelevant ones, based on the probability or the RSV values assigned. Threshold adjustment helps to achieve a utility target for the selected documents. These are performed periodically after a number of documents have gone through the process.

Initial profile is defined using the raw topic and the three judged relevant documents from the training set. Once the filtering process begins, statistics of term usage is kept for all documents passing through. Moreover, for the documents selected, whether relevant or not, they are identified as a separate retrieval collection for threshold adjustment. We re-compute the RSV of those documents based on the current profile and then adjust the threshold to provide us with the maximum utility in regard to the filtered documents. We then use that threshold to filter future incoming documents.

As more relevant documents are selected, we expand the profile by adding terms that have higher frequency in the filtered relevant. A maximum of 30 is set as a limit for the number of expanded terms.

We also keep track of precision values, both the global and local ones. Global precision is the precision from the start of the filtering to the current point while the local one contains only the precision for the last two update cycles. We think relevant documents are not distributed uniformly over the course of time but are clustered over certain regions in the timeline of the document stream. If the current local precision is significantly higher than the global one, we feel that we are in a region with relevant documents clustered and the filtering threshold should be lowered so that more relevant documents can be selected. On the other hand, if the global precision is significant higher. It means we are in a region where very few documents are relevant and one should tighten the threshold so that fewer irrelevants will be selected.

Lastly, a query term co-occurrence filtering method was implemented in addition to statistical filtering to aim at achieving better precision. Query term pairs were formed from the original topic using the title or description fields. During filtering, the presence of a query term pair in a document sentence is considered as evidence for selection even if RSV is somewhat less than the current threshold. Assume the current RSV threshold is T. Normally documents with RSV > T will be selected for the user. This is now modified as follows:

```
If (docRSV >= 1.5*T  OR  (0.9*T < docRSV
        < 1.5*T  &&   has-co-occurrence))
        select-document;
else  reject-document;
```

Results and Discussion

We submitted 4 runs pirc2F{01,02,03,04}. pirc2F03 and pircs2F04 are base runs without phrase filtering but using different initial parameters. pirc2F01 and pirc2F02 are based on pirc2F03 but with phrase filtering using a window of three sentences and whole document respectively. Results were not good, especially for the intersection topics. For example, the better run is pirc2F01 with mean scaled T11U = 0.154 for the 50 assessor topics and 0.047 for the 50 intersection topics. Phrase filtering seems useful compared to not using it: average score for the two base runs is only about half of the two runs with phrase filters. The experimental results were low and we suspect programming bugs in some of our procedures.

5 Conclusion

We proposed an approach to finding answer pages for topic distillation in a collection of web documents based on the properties of 'key resource': emphasis on content, link information and host diversity in answer list. In novelty task, we employ a large dictionary with TREC-8 statistics to aid our retrieval with short sentences, and WordNet to help expand words with synonyms for evaluating similarity among sentences. A phrase filtering procedure was tested for the adaptive filtering task.

Acknowledgments

This work was partially supported by the Space and Naval Warfare Systems Center San Diego, under grant No. N66001-1-8912.

References

1. Hawking, D & Craswell, N (2002). Overview of the TREC-2001 Web Track. In: Information Technology: The Tenth Text Retrieval Conference, TREC 2001. NIST SP 500-250. pp.61-67.

2. Chakrabarti, S, Dom, B, Raghavan, P, Rajagopalan, S, Gibson, D. & Kleinberg, J. (1998a). Automatic resource compilation by analyzing hyperlink structure and associated text. Proc. 7th WWW Conf. pp.65-74.

3. Chakrabarti, S, Dom, B. S, Gibson, D. Kumar, R. Raghavan, P, Rajagopalan & Tomkins, A. (1998b). Experiments in topic distillation. ACM SIGIR'98 Post Conf. Workshop on Hypertext IR for the Web.

4. Bharat, K & Henzinger, M.R (1998). Improved algorithm for topic distillation in a

hyperlinked environment. Proc. ACM SIGIR 1998, pp.104-111.

5. Amento, B, Terveen, L & Hill, W (2000). Does "authority" mena quality? Predicting expert quality ratings of web documents. Proc. ACM SIGIR 2000, pp.296-303.

6. Kleinberg, J. (1998). Authoritative sourcs in a hyperlinked environment. In: Proc. of 9th ACM-SIAM Symposium on Discrete Algorithms.

7. Chowdhury, A, Frieder, O, Grossman, D & McCabe, M.C (2002). Collection statistics for fast duplicate document detection. ACM TOIS 20 pp.171-191.

8. Allan, J, Carbonell, J, Doddington, G & Yamron, J (2001). Topic detection and tracking pilot study.

9. Zhang, Y, Callan, J & Minka, T (2002). Novelty and redundancy detection in adaptive filtering. Proc. ACM SIGIR 2002, pp.81-88.

Shot boundary detection using the moving query window

S.M.M. Tahaghoghi James A. Thom Hugh E. Williams

School of Computer Science and Information Technology
RMIT University, GPO Box 2476V
Melbourne, Australia, 3001

{saied,jat,hugh}@cs.rmit.edu.au

Abstract

The large volume of video content generated each day requires efficient and effective methods of video indexing and retrieval. A common first step in indexing video content is to identify visually and semantically continuous segments or shots. In this paper, we present the moving query window approach to video shot boundary detection. This uses the techniques of query-by-example (QBE) and ranked results, both often used in content-based image retrieval (CBIR). Each frame of the video is used in turn as an example query on the image collection formed by the other frames within a moving window. Transitions are detected by monitoring the relative ranks of these frames in the results list. We show that this is an effective approach for the shot boundary detection task of the TREC-11 video track.

1 Introduction

Video is the next frontier of visual information retrieval: for archived footage to be useful, its contents must be known. Since a video clip has a time dimension, this generally means that the content must be reviewed sequentially, and sections of interest identified. This is costly and tedious to perform manually, and so automatic techniques are required.

A video stream can be considered to be composed of small, coherent sections or *shots*, where adjacent frames are usually similar. A small sample of frames can be selected from each shot and indexed for use in video retrieval [4, 21]. The answer to a video retrieval information need is then a list of shots containing frames similar to the query requirements.

A shot is bounded at each end by a transition. The main types of transition are cuts, fades, dissolves, and spatial edits [5, 16]. The type and frequency of transitions in a video clip is largely dependent on the age of the footage and the nature of the content. Almost all transitions in fast-moving television news footage are cuts, and dissolves are rare. In a documentary, dissolves and fades appear frequently. Cuts, dissolves, and fades account for the majority of transitions; Lienhart [10] reports this is more than 99%, and similar ratios were observed in the TREC-10 and TREC-11 video collections.

Video footage can be segmented into shots by detecting the shot start and end points, as signified by transitions. The difference between adjacent frames of a shot is usually small, but increases during transitions. Most shot boundary detection algorithms identify transitions by monitoring for significant changes in the video frames.

One method to measure this change is to compare frames pixel-by-pixel: transitions are reported if the colour or intensity of a significant number of pixels changes between frames [2]. However, pixel-by-pixel comparison of frames is generally computationally intensive, and sensitive to object motion, noise, camera motion, and changes in camera zoom. Using information produced during video compression is typically efficient [1, 12, 23], but may result in low precision [2]. Computing and comparing statistics of the frames—such as the mean and standard deviation of pixel values [9], or histograms of colour usage [13, 25]—reduces sensitivity but adds computation overhead.

Recent work in this area using colour histograms includes that of Pickering et al. [14]. In their approach, frames are divided into nine blocks, and red, green, and blue (RGB) colour component histograms extracted from each. The Manhattan distance between the histograms of corresponding blocks is calculated, and the

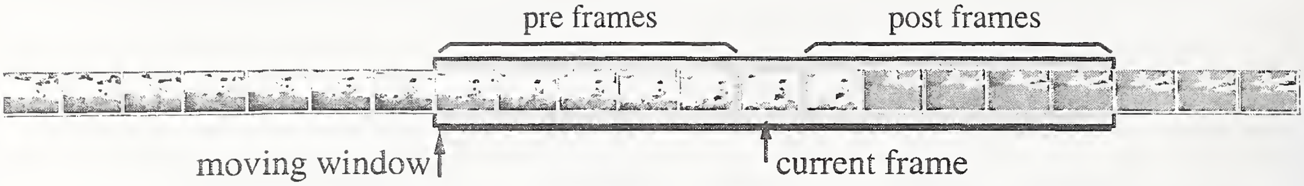


Figure 1: Moving query window with a half window size (HWS) of 5; the five frames before and the five frames after the current frame form a collection on which the current frame is used as a query example.

largest of the three is retained as the distance between the blocks. The median of the nine individual inter-block distances is taken as the inter-frame distance. A transition is reported if this distance is greater than a fixed threshold and also greater than the average distance value for the 32 surrounding frames.

Sun et al. [18] compare the colour histograms of adjacent frames within a moving window; a shot boundary is reported if the distance between the current frame and the immediately preceding one is the largest inter-frame distance in the window, and significantly larger than the second largest inter-frame distance.

The IBM `CueVideo` program uses a sampled three-dimensional RGB colour histogram to measure the distance between pairs of frames [17]. Histograms of recent frames are cached, and statistics are calculated for this moving window. These statistics are used to determine adaptive threshold levels.

In this paper, we present our approach to video segmentation based on the concepts of querying by example image (QBE) and ranked results, both regular features of content-based image retrieval (CBIR). In the next section, we introduce our new approach. Section 3 addresses our choice of features and parameters. In Section 4, we review the performance of our technique on the TREC-11 shot boundary detection task. In Section 5, we conclude and discuss possible areas for improvement.

2 The Moving Query Window Technique

A content-based image retrieval or CBIR system aims to satisfy the information need of a user by selecting images from the collection that best meet the user's requirements. With many CBIR systems, users convey their requirements by selecting features such as colour and texture from a palette [3], sketching a representation of the desired image [8], or providing an example image that captures the qualities of the target image [7, 19]. The last two methods are categorised as query-by-example, or QBE.

In CBIR, a summary is produced for each image in the collection that captures visual aspects such as colour and texture distributions, and the shape and location of objects in the image. When using QBE, a corresponding summary is produced for the query. These summaries are compared, and collection images are ranked by similarity to the query. The user is then presented a list of all the images in the collection, ranked from most- to least-similar.

We have applied the concepts of QBE and ranking to the video segmentation problem. We were motivated to explore this approach by the observation that frames preceding a transition are similar in content but are usually dissimilar to those following the transition.

We define a moving window of size N extending equally on either side of the current frame, but not including the current frame itself. The number $\frac{N}{2}$ is referred to as the half window size (HWS). We refer to the $\frac{N}{2}$ frames preceding the current frame as the *pre-frames*. Similarly, the $\frac{N}{2}$ window frames following the current frame are *post-frames*. Figure 1 shows a moving window of ten frames, with five pre- and post-frames on either side of the current frame.

We use the current frame as a query on the collection of frames inside this moving window, that is, to the pre- and post-frames. This QBE orders the N collection frames by decreasing similarity to the query frame, with the most similar frame first, and the most dissimilar frame last.

The difference between the current frame—which is used as the query example—and the frames before and after it will usually be near-symmetrical. Thus, the pre- and post-frames will be interspersed throughout the ordered list of window frames, and the number of pre- and post-frames in the top $\frac{N}{2}$ results will be approximately equal. However, this changes in the vicinity of a transition.

Pre-frames	Current frame	Post-frames	NumPreFrames
A A A A A A A A A A	A	A A A A A A A A A A	5
A A A A A A A A A A	A	A A A A A B B B B B	7
A A A A A A A A A A	A	B B B B B B B B B B	10
A A A A A A A A A A	B	B B B B B B B B B B	0
A A A A A A B B B B	B	B B B B B B B B B B	2

Figure 2: As the moving window traverses an abrupt transition, the number of pre-frames in the $\frac{N}{2}$ frames most similar to the current frame varies significantly. This number (`NumPreFrames`*) rises to a maximum just before an abrupt transition, and drops to a minimum immediately afterwards.*

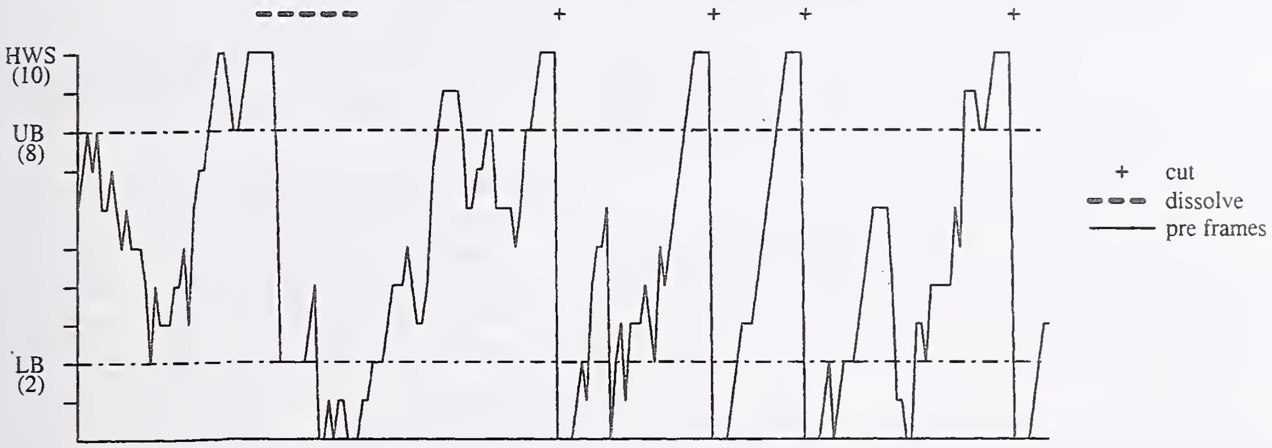


Figure 3: Plot of the number of pre-frames in the top half of the ranked results for a 200-frame interval. The five transitions present in this interval are indicated above the plot. The parameters used for HWS, the upper threshold (UB) and the lower threshold (LB) are listed between parentheses.

2.1 Abrupt Transitions

As the current frame approaches a cut, frames from the second shot enter the window. All the pre-frames are from the first shot (shot A), while some of the post-frames belong to the second shot (shot B). However, the current frame is still from shot A, so after computing the similarity to the query, we generally find the shot B frames ranked lower than the shot A frames. As a result, there is a rise in the number of pre-frames in the top $\frac{N}{2}$.

When the current frame is the last frame of shot A, all pre-frames are from shot A, and all post-frames are from shot B. At this point, the number of pre-frames in the top $\frac{N}{2}$ reaches a maximum, since the shot A frames will all be ranked above the shot B frames. This can be seen in Figure 2. Once the current frame moves into the next shot, the example image will belong to shot B, so the situation is reversed: the number of post-frames in the top $\frac{N}{2}$ exhibits a sharp rise, while the number of pre-frames drops to near zero.

In Figure 3, the variation in the number of pre-frames in the top $\frac{N}{2}$ results is shown over 200 frames of a clip. The location of the four cuts and one dissolve in this interval are shown at the top of the figure. Cuts are accompanied by a sharp drop in the number of pre-frames at the top of the ranked list.

2.2 Gradual Transitions

The first transition in Figure 3 is a gradual transition. We see that frame ranks within the moving window are also affected by this transition, although to a lesser degree than by the cuts. Our technique can be modified to additionally detect gradual transitions. When the moving window traverses a gradual transition, we observe three phases:

1. **Post-frames enter transition, but the current frame is not yet in transition:** The number of pre-frames ranked in the top $\frac{N}{2}$ rises, since the transition frames are less similar to the example frame than the non-transition frames.
2. **Current frame in transition:** The number of pre-frames ranked in the top $\frac{N}{2}$ slowly decreases.
3. **Current frame exits transition:** The number of pre-frames ranked in the top $\frac{N}{2}$ falls significantly, since the pre-frames—which are still within in the transition—are less similar to the example than the post-frames.

The three phases of this transition can be seen from the plot at the top of Figure 3. Considering the number of pre-frames in the top $\frac{N}{2}$ results, we see that this number increases towards the peak as we approach the start of a transition. During the transition, the number returns to moderate values. As the current frame exits the transition, the number of pre-frames drops to a minimum; the value gradually increases again as the transition frames leave the half-window preceding the current frame. We can detect gradual transitions by monitoring for this characteristic pattern.

In general, detection of gradual transitions is more difficult than detection of abrupt transitions. In contrast to cuts, gradual transitions do not have a sharp division between shots, and adjacent frames within a gradual transition usually differ by a small amount. To accentuate the differences between the frames, we could sample the stream at a lower rate. This would, however, reduce our precision: if we use every nth frame, we can only resolve the shot boundary to within n frames.

In our experiments, we use all frames, employing each in turn as a query example. However, we omit the closest few frames bordering the current frame from the collection. This leaves a gap, which we refer to as the *Demilitarised Zone* (DMZ) on either side of the current frame, as illustrated in Figure 4. The DMZ effectively determines the difference between the example frame and the most similar frame from the window; a large value for the DMZ will blur the distinction between frames of shot A and frames of shot B.

2.3 Algorithm Details

In this section, we describe the details of our shot boundary detection scheme. We begin by defining the algorithm parameters, and continue with a description of the detection steps for transitions.
In our discussion, we refer to four primary parameters:

Lower Bound (LB): This is the lower threshold. Once the number of pre-frames falls below this level, a possible transition is detected as shown in Figure 3.

Upper Bound (UB): This is the upper threshold. Once the number of pre-frames rises above this level, a possible transition is detected as shown in Figure 3.

Half Window Size (HWS): The number of frames from either side of the current frame that are contained within the moving window. Since we examine the top $\frac{N}{2}$-ranking frames, we use this number as the main parameter, rather than the full window size (N) itself. This is shown in Figure 4.

Demilitarised zone depth (DMZ): This is the size of the gap between the current frame and the nearest frame that is part of the moving window. See Figure 4 for an example.

We continue next with a discussion of how abrupt transitions are detected using the moving window and these parameters.

Detection of Cuts

To detect abrupt transitions, we monitor the number of pre-frames in the top $\frac{N}{2}$ results as each frame is examined. We refer to this number as `NumPreFrames`. We also measure the slope of the `NumPreFrames` curve. This is normally small, that is, around two.

As we near an abrupt transition, `NumPreFrames` rises quickly and passes the upper bound (UB). Once we pass the transition, `NumPreFrames` falls sharply below the lower bound (LB). The slope reflects this by taking on a large positive value, followed quickly by a large negative value. This behaviour can be observed in

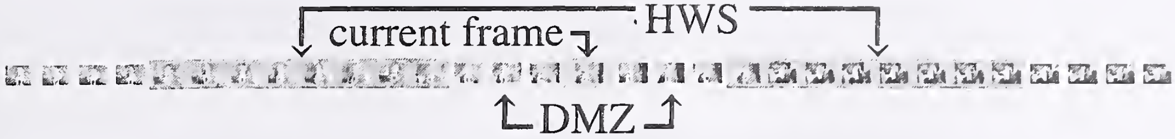


Figure 4: Moving query window with a half window size (HWS) of 8, and a demilitarised zone (DMZ) of three frames on either side of the current frame; the eight frames preceding and the eight frames following the current frame form a collection, against which the current frame is used as a query example.

Figure 3. We report a possible cut if NumPreFrames exceeds UB, then falls below LB in the space of two frames.

In some cases, the slope condition may be satisfied inside a shot where no transition exists. This may occur where, for example, a traffic light changes from red to green; all "red" frames will be ranked together and separately from all "green" frames, causing the slope to exhibit the behaviour seen for cuts. To avoid incorrectly declaring a cut in such cases, we impose the condition that there must be a large difference between the pre- and post-frames. This is achieved by requiring the average distance of the top $\frac{N}{2}$ frames to the query image to be less than half the average distance of the bottom $\frac{N}{2}$ frames from the same query image.

All comparisons so far have been relative. To further reduce the occurrence of false positives, we introduce an absolute threshold for the distance between the last pre-frame and the first post-frame. This is expressed as a proportion of the maximum distance possible between two frames using the current feature and histogram representation. We fixed this threshold at 25% of the maximum possible distance.

To summarise, a cut is reported if the following conditions are satisfied:

1. The NumPreFrames slope takes on a large negative value;
2. The top $\frac{N}{2}$ frames are significantly different from the bottom $\frac{N}{2}$ frames; and,
3. The last pre-frame and the first post-frame are significantly different.

Since these conditions are not synchronous, we allow them to be satisfied at any point within an interval of four frames. For example, the first condition may be met at frame n, and the second condition may be met at frame $n+2$. If all three conditions hold, we record a cut with the current frame being the first frame of the new shot.

Detection of Gradual Transitions

Detection of gradual transitions is more difficult than detection of abrupt transitions, and we need to employ more heuristics. We experimented only briefly with gradual transition detection in this work and, as we show later, our detection of gradual transitions is relatively ineffective. We plan further experiments to determine the variation of parameters required for improved detection of such transitions.

We noted in Section 2.2 that during a gradual transition, NumPreframes often rises to high levels, then drops to low values, and remains there for a some time before rising to return to typical levels. We are alerted to a possible gradual transition when we detect that NumPreframes has remained low for several frames. We regard the current frame as marking the end of the transition. To identify the beginning of the transition, we look back to find the location of the first phase of the gradual transition, that is, the point where NumPreFrames first rises to a high level designated by the upper bound (UB). Finally, we measure how long NumPreframes remains high. If this is more than a threshold value, we declare a gradual transition.

In summary, a gradual transition is reported if the following conditions are met:

1. The NumPreFrames slope remains low for several frames, and
2. before this, NumPreFrames increases to a high level, and remains consistently high over several frames.

If both conditions are met, we record a gradual transition starting at the point NumPreFrames first exceeds the upper bound, and ending at the current frame.

3 Selection of Features and Parameters

To compare different features and identify suitable parameters, the moving query window algorithm was ap-

Parameter	Acronym	Range start	Range end	Step size
Half window size ($\frac{N}{2}$)	HWS	6	30	2
Lower bound	LB	1	4	1
Upper bound	UB	$HWS-4$	$HWS-1$	2
De-militarised zone	DMZ	0	10	2

Table 1: The ranges of values used for the parameters of the shot boundary detection algorithm.

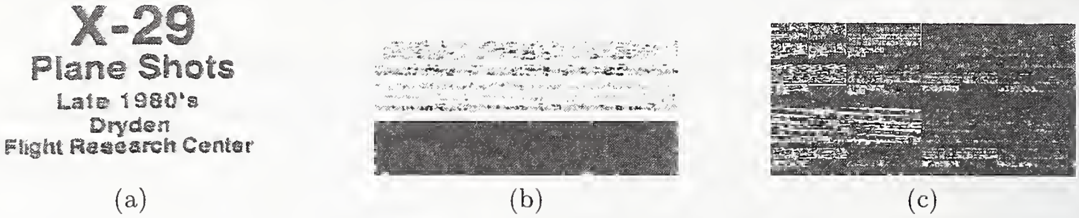


Figure 5: (a) Input frame of dimensions 352×240. (b) Frame Y (brightness) data placed in a super-frame of dimensions 512×256, with the unused portion of the super-frame being set to black. (c) Transformed super-frame; the data corresponding to the unused portion of the super-frame does not contain any information, and is discarded.

plied to detect shot boundaries on a subset of the TREC-10 evaluation set comprising eleven clips, containing a total of 996 cuts and 406 gradual transitions. Each feature was evaluated using parameters in the ranges shown in Table 1.

The effectiveness of the segmentation operation is evaluated using the standard information retrieval measures of recall and precision. Precision represents the fraction of detected transitions that match the reference data:

$$P = \frac{\text{Transitions correctly reported}}{\text{Total transitions reported}}$$

Recall measures the fraction of all reference transitions that are correctly detected:

$$R = \frac{\text{Transitions correctly reported}}{\text{Total reference transitions}}$$

These two measures can be used for both abrupt and gradual transitions. To evaluate how well reported gradual transitions overlap with reference transitions, TREC-11 introduced the measures *Frame Precision* (*FP*) and *Frame Recall* (*FR*).

$$FP = \frac{\text{Frames correctly reported in detected transition}}{\text{Frames reported in detected transition}}$$

$$FR = \frac{\text{Frames correctly reported in detected transition}}{\text{Frames in reference data for detected transition}}$$

3.1 Features

We used one-dimensional global histograms using the HSV, Lab, and Luv colour spaces, and a fourth feature derived from the Daubechies wavelet transform of the frames. Preliminary experiments using three-dimensional colour histograms have produced slightly better results but we do not describe them here.

The native colour space of the MPEG compressed video stream is YC_bC_r. The wavelet-based feature for each frame was generated by computing the six-tap Daubechies wavelet transform coefficients from the YC_bC_r colour data. When calculating the wavelet transform using the Mallat algorithm, the data dimensions are halved after each pass [11, 22]. Thus, we can perform four passes on frames with dimensions 352×240, ending at 22×15, which cannot be transformed further. Frames with dimensions 320×240 can also be transformed four times (ending at 20×15), while frames with dimensions 352×288 can be transformed five times (ending at 11×9).

All clips used in TREC-11 had dimensions 352×240; nevertheless, we should cater for different frame sizes. To allow comparison of equivalent wavelet scales for different-size frames without the expense of resizing, we rearrange the frame data to fit into a *super-frame* with dimensions that are a power of two. For example, the pixel data from a 352×240 frame is inserted into a super-frame of dimension 512×256, as shown in Figure 5. The unused portion of the super-frame

Abrupt	Bins/subbands	HWS	LB	UB	DMZ	Gradual	Bins/subbands	HWS	LB	UB	DMZ
HSV	384	20	4	18	0	HSV	48	20	4	18	4
Lab	1536	26	3	24	4	Lab	192	22	3	20	4
Luv	1536	10	4	8	0	Luv	1536	22	3	20	4
RWav	5	10	3	8	4	RWav	4	20	3	18	4

Table 2: The best set of parameters varies by feature and transition type; gradual transitions are generally best detected with a DMZ of four. The effect of varying the DMZ is less pronounced for cut detection. While in some cases the best results are obtained with non-zero DMZ, the difference with the DMZ=0 results is insignificant.

is zero-filled, and the transform data for this portion is later discarded. With the new frame dimensions, eight transform passes are possible, ending with the data dimensions 2×1. We call this feature the wavelet transform on re-ordered data or RWav.

Of the feature combinations tried, RWav proved to be the most effective for detecting cuts, and Luv was the best feature to use for detecting gradual transitions. The simple HSV feature also proved to be effective, with recall and precision comparable to those of the best features. The amount of processing required to extract the HSV data from the video stream is much less than the other features. This low extraction cost may make HSV the most practical choice of feature for a commercial system.

We found that while using only the luminance component of the colour data trebles processing speed, detection effectiveness is significantly reduced. An exception is the RWav feature, where the effectiveness in detecting cuts with only luminance (Y) information is relatively unchanged from the full YC_bC_r version.

Although the global colour features generally produced good results, they often failed to detect cuts between two shots of the same scene where the camera followed an object moving rapidly against a noisy background. This type of cut is often easily detected by the wavelet (RWav) feature, which preserves spatial layout information.

Conversely, the wavelet feature is sensitive to small changes in the frame content and performs relatively poorly at finding gradual transitions. However, the high-frequency data—corresponding to detail in the image—plays an important part in cut detection; we observe the best results when using the first four or five transform sub-bands. Further increasing the number of sub-bands inserts too much detail, and adversely affects performance. The volume of feature data stored per frame also quadruples for each additional sub-band, and so a performance penalty is also incurred.

3.2 Other Parameters

The best choice of algorithm parameters varied for different features and for the two transition types; these are listed in Table 2.

We found that transitions are best detected with a half window size (HWS) of approximately 18 or 20 frames. It is likely that the optimal value for HWS will vary depending on the content of the footage being examined; long, slow transitions will favour larger values of HWS. We have not performed in-depth experiments to test this supposition.

The lower bound (LB) and upper bound (UB) determine the relative priorities of recall and precision. Decreasing LB towards zero generally increases precision at the cost of recall. This effect is relatively minor for cut detection, since in most cases, `NumPreFrames` actually reaches zero at the cut boundary. Detection of gradual transitions is sensitive to the LB parameter, and our best preliminary results were obtained with an LB of 3 or 4.

There is a close relationship between the best choice of frame gap (DMZ) and the type of transition to be detected. Cuts are generally best detected with no gap at all (DMZ=0), while gradual transitions are best found with a small gap (DMZ=4). As with HWS, we believe the best value is somewhat dependent on the type of video footage being processed and we plan further experiments to verify this.

Histogram distances were calculated using the Manhattan, cumulative Manhattan, histogram intersection, and Euclidean measures. The Manhattan distance measure produced the best results. The relatively high computation cost of the Euclidean distance measure makes it unattractive for use in video.

4 TREC-11 Results

In TREC-11, groups were permitted to submit a maximum of ten runs. The evaluation set consisted of eighteen video clips, with 1 466 cuts and 624 gradual transi-

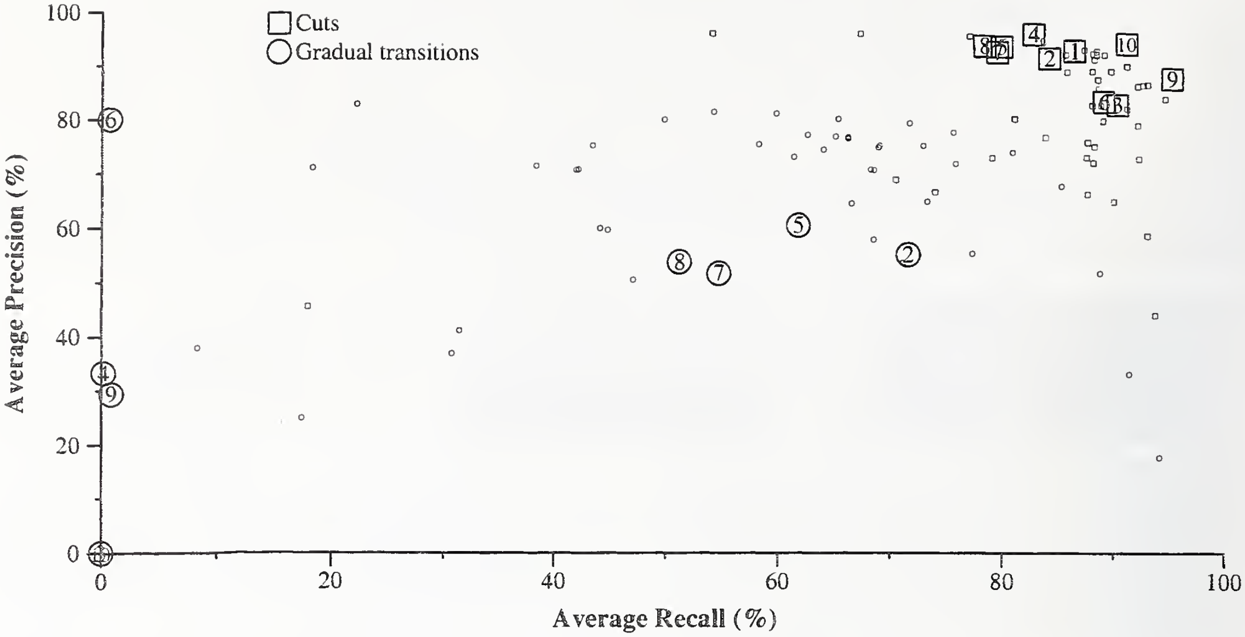


Figure 6: Performance of the moving query window for cuts and gradual transitions on the TREC-11 shot boundary detection task.

Run	Feature type	Colour space	Vector length	Half window size (HWS)	Lower Bound (LB)	Upper Bound (UB)	Demilitarised Zone (DMZ)
1	Colour histogram	HSV	384	20	4	18	0
2	Colour histogram	HSV	96	20	3	16	4
3	Colour histogram	Lab	1536	12	6	10	0
4	Colour histogram	Lab	1536	26	3	24	0
5	Colour histogram	Lab	1536	26	3	24	4
6	Colour histogram	Luv	1536	10	4	8	0
7	Colour histogram	Luv	1536	22	3	20	4
8	Colour histogram	Luv	1536	26	3	24	4
9	Wavelet (5 scales)	YC_bC_r	1176	10	3	8	0
10	Wavelet (5 scales)	YC_bC_r	1176	20	3	18	0

Table 3: Parameters used for each submitted run.

tions. We submitted runs using the parameters shown in Table 3.

The recall and precision levels for cuts and gradual transitions are shown in Figure 6. The numbered squares and numbered circles correspond to moving query window results for abrupt and gradual transitions respectively. Results submitted to TREC-11 by other groups are indicated by the small squares and circles. Similarly, Figure 7 shows the performance of our approach and that of other systems when detecting gradual transitions, as measured by Frame Recall and Frame Precision.

The moving query window showed good results for cut detection and poor results for gradual transitions. Algorithm parameters that performed well on abrupt transitions performed poorly on gradual transitions and vice-versa. Run ten produced the best results for detection of abrupt transitions, but failed to detect any gradual transitions.

5 Summary

We have introduced a new moving query window approach that applies the CBIR concepts of querying

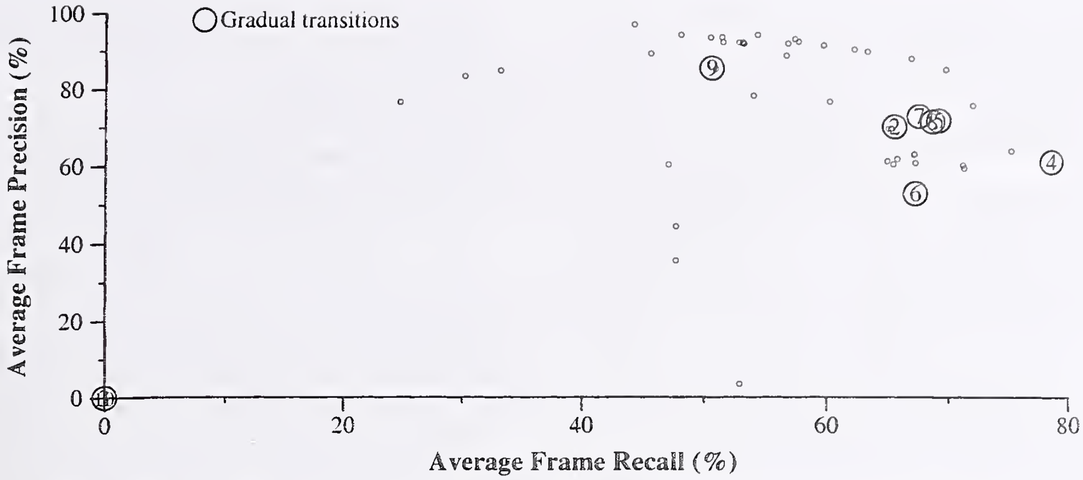


Figure 7: Performance of the moving query window for gradual transitions on the TREC-11 shot boundary detection task, as measured by Frame Recall and Frame Precision.

by example image and ranked results to detect shot boundaries in video. We have described the parameters of the algorithm, and discussed the steps used to determine the presence of transitions.

We have identified several areas where modifications could lead to improved efficiency and effectiveness. One improvement could be to preserve some information about the spatial colour distribution in the colour features; this can be done by using local rather than global colour histograms.

Our algorithm is sensitive to sudden changes in the video brightness level, photographic flashes, and the appearance and disappearance of textual captions. This sensitivity can be reduced by integrating existing work on detectors for such phenomena [15, 24].

Populating the window requires that the algorithm begin operation from the $\frac{N}{2}$th frame, and end $\frac{N}{2}$ frames before the end. Transitions occurring within the excluded regions cannot be detected. Other methods must be used to handle the approximately half-second of footage at the extremities of each clip.

The routines for detection of cuts and gradual transitions are independent, and may interfere destructively. Since the conditions to be met for cuts are stricter than those for gradual transitions, we made a decision to give precedence to cuts; if a cut has already been detected in the transition interval, the gradual transition is not reported. In addition, we have not experimented in detail with the detection of gradual transitions; we plan future work on selecting heuristics for this domain.

Overall, we have shown that our method produces competitive results. In particular, we have shown that the RWav feature, derived from the Daubechies wavelet transform of the frame data, produces excellent cut detection results. Our parameters were based on experiments using a subset of the TREC-10 evaluation set, and are therefore not necessarily optimal for the TREC-11 evaluation set. We expect that results can be improved through experimentation with dynamic thresholds and other adaptive parameters.

References

[1] F. Arman, A. Hsu, and M.-Y. Chiu. Image processing on encoded video sequences. *Multimedia Systems*, 1(5):211–219. Springer-Verlag, March 1994.

[2] J. S. Boreczky and L. A. Rowe. Comparison of video shot boundary detection techniques. *Journal of Electronic Imaging*, 5(2):122–128. SPIE, April 1996.

[3] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz. Efficient and effective querying by image content. *Journal of Intelligent Information Systems*, 3(3/4):231–262. Kluwer, July 1994.

[4] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, and D. Steele. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32. September 1995.

[5] A. Hampapur, R. Jain, and T. Weymouth. Digital video segmentation. In *Proceedings of the ACM In-*

ternational Conference on Multimedia, pages 357–364, San Francisco, California, USA, 15–20 October 1994.

[6] D. Harman. Overview of the Second Text REtrieval Conference (TREC-2). *Information Processing & Management*, 31(3):271–289. Elsevier, May/June 1995.

[7] Y. Ishikawa, R. Subramanya, and C. Faloutsos. Mindreader: Querying databases through multiple examples. In *Proceedings of the International Conference on Very Large Data Bases (VLDB'98)*, pages 218–227, New York, USA, 24-27 August 1998.

[8] C. E. Jacobs, A. Finkelstein, and D. H. Salesin. Fast multiresolution image querying. In *Proceedings of the ACM-SIGMOD International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH'95)*, pages 277–286, Los Angeles, California, USA, 6–11 August 1995.

[9] R. Kasturi and R. Jain. *Dynamic Vision*. In *Computer Vision: Principles*, pages 469–480. IEEE Computer Society Press, 1991.

[10] R. W. Lienhart. Comparison of automatic shot boundary detection algorithms. *Proceedings of the SPIE; Storage and Retrieval for Still Image and Video Databases VII*, 3656:290–301. December 1998.

[11] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693. July 1989.

[12] J. Meng, Y. Juan, and S.-F. Chang. Scene change detection in a MPEG compressed video sequence. *Proceedings of the SPIE; Digital Video Compression: Algorithms and Technologies*, 2419:14–25. April 1995.

[13] A. Nagasaka and Y. Tanaka. Automatic video indexing and full-search for video appearances. *Visual Database Systems*, 2:113–127. Elsevier, 1992.

[14] M. Pickering and S. M. Rüger. Multi-timescale video shot-change detection. In *NIST Special Publication 500-250: Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pages 275–278, Gaithersburg, Maryland, USA, 13–16 November 2001.

[15] G. Quénot and P. Mulhem. Two systems for temporal video segmentation. In *Proceedings of the European Workshop on Content Based Multimedia Indexing (CBMI'99)*, pages 187–194, Toulouse, France, 25–27 October 1999.

[16] A. Smeaton, P. Over, and R. Taban. The TREC-2001 video track report. In *NIST Special Publication 500-250: Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pages 52–60, Gaithersburg, Maryland, USA, 13–16 November 2001.

[17] J. R. Smith, S. Srinivasan, A. Amir, S. Basu, G. Iyengar, C. Y. Lin, M. R. Naphade, D. B. Ponceleon, and B. L. Tseng. Integrating features, models, and semantics for TREC video retrieval. In *NIST Special Publication 500-250: Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, pages 240–249, Gaithersburg, Maryland, USA, 13–16 November 2001.

[18] J. Sun, S. Cui, X. Xu, and Y. Luo. Automatic video shot detection and characterization for content-based video retrieval. *Proceedings of the SPIE; Visualization and Optimisation Techniques*, 4553:313–320. September 2001.

[19] S. M. M. Tahaghoghi, J. A. Thom, and H. E. Williams. Multiple-example queries in content-based image retrieval. *Proceedings of the Ninth International Symposium on String Processing and Information Retrieval (SPIRE'2002)*, pages 227–240, Lisbon, Portugal, September 2002.

[20] Text REtrieval Conference (TREC), National Institute of Standards and Technology, Gaithersburg, Maryland, USA.
URL: `http://trec.nist.gov`.

[21] S. Uchihashi, J. Foote, A. Girgensohn, and J. Boreczky. Video manga: Generating semantically meaningful video summaries. In *Proceedings of the ACM International Conference on Multimedia*, pages 383–392, Orlando, Florida, USA, 30 October – 5 November 1999.

[22] J. R. Williams and K. Amaratunga. Introduction to wavelets in engineering. *International Journal for Numerical Methods in Engineering*, 37(14):2365–2388. John Wiley & Sons, 1994.

[23] B. L. Yeo and B. Liu. Rapid scene analysis on compressed video. *IEEE Transactions on Circuits and Systems for Video Technology*, 5(6):533–544. December 1995.

[24] D. Zhang, W. Qi, and H. J. Zhang. A new shot boundary detection algorithm. *Lecture Notes in Computer Science; Proceedings of the Second IEEE Pacific Rim Conference on Multimedia (PCM'2001)*, 2195:63–70. Beijing, China, 24–26 October 2001.

[25] H. J. Zhang, A. Kankanhalli, and S. W. Smoliar. Automatic partitioning of full-motion video. *Multimedia Systems*, 1(1):10–28. Springer-Verlag, June 1993.

Rutgers Interactive Track at TREC 2002

N.J. Belkin, C. Cool*, D. Kelly, G. Kim, J.-Y. Kim, H.-J. Lee, G. Muresan, M.-C. Tang, X.-J. Yuan
School of Communication, Information & Library Studies, Rutgers University and
*Graduate School of Library and Information Studies, Queens College, CUNY
[belkin | diane | gkim | jaykim | hyukjinl | muresan | muhchyun | xjyuan]@scils.rutgers.edu *ccool@qc.edu

1 Introduction

Two important results came out of our investigations in the TREC 2001 Interactive Track (Belkin, et al, 2002). One was that the greater the amount of interaction that searchers engaged in, the lower their satisfaction with the results of the search. We understood this to mean that interaction effort was inversely related to search satisfaction, and therefore, that making interaction more effective would lead to increased search satisfaction. The second was that performance in the searching task increased with query length. We conjectured that this was due, at least in part, to the subjects having searched using a best-match search engine (Excite[1]), as well as longer queries being better able to express the information problem. These two findings became the basis for our systems and experiments in the TREC 2002 Interactive Track. We formed the following hypotheses:

1. A system designed to reduce the amount of interaction that a searcher has to engage in, by making it more effective, will lead to increased satisfaction with search results, and increased performance, as compared to a system not so designed;
2. A system which encourages long qucries will lead to better performance in the search task than one which does not.

In order to test the first hypothesis, we designed two basic interfaces to the Panoptic search engine[2]: one which presented the results of a query as a ranked list of titles of documents, twenty at a time; the other which presented the results of a query as the texts of four documents at a time, each in a scrollable window, ranked in the same order as the first interface. The second interface was intended to reduce user interaction with the system by virtue of not requiring the searcher to follow links from the search results to the actual documents and then back again to the results list, as in the first interface. It was also thought that being able to see the documents immediately would make it easier and faster to evaluate their potential relevance to the search topic, than having first to evaluate on the basis of a title plus snippet surrogate, and then do a second evaluation based on the page itself.

To test the second hypothesis, we designed two different query elicitation methods, that were used in both interfaces. One method had just the word "query" above the box in which the query was to be entered. When this version of either interface was demonstrated to the subjects in the experiment, the experimenter would enter the query as a list of words and phrases. The second method had, above the query entry box, the following: "Information problem description (the more you say, the better the results are likely to be)". When this version of the interfaces was demonstrated, the experimenter entered one or more complete sentences or questions descriptive of the topic and desired results. The second condition was predicted to lead to longer queries, both in terms of all of the words entered, and in terms of the words that were finally interpreted by the Panoptic engine, which used a stop list.

Of course, the treatments which we designed were themselves only predicted to have the desired results. Therefore, in order to investigate the hypotheses, it was first necessary to determine whether these different treatments did in fact lead to the desired results, i.e. less interaction and longer queries. In a sense, then, the specific treatments were hypotheses themselves, which we also investigated. This paper therefore presents results with respect to hypotheses 1 and 2, above, and with respect to the following hypotheses:

3. A search interface which directly presents the ranked documents retrieved by a search will lead to less user-system interaction than one which presents ranked titles and requires following links to view documents;
4. A search interface which asks searchers to describe their information problems at length will lead to longer queries than one which asks searchers to simply input a query as a list of words or phrases.

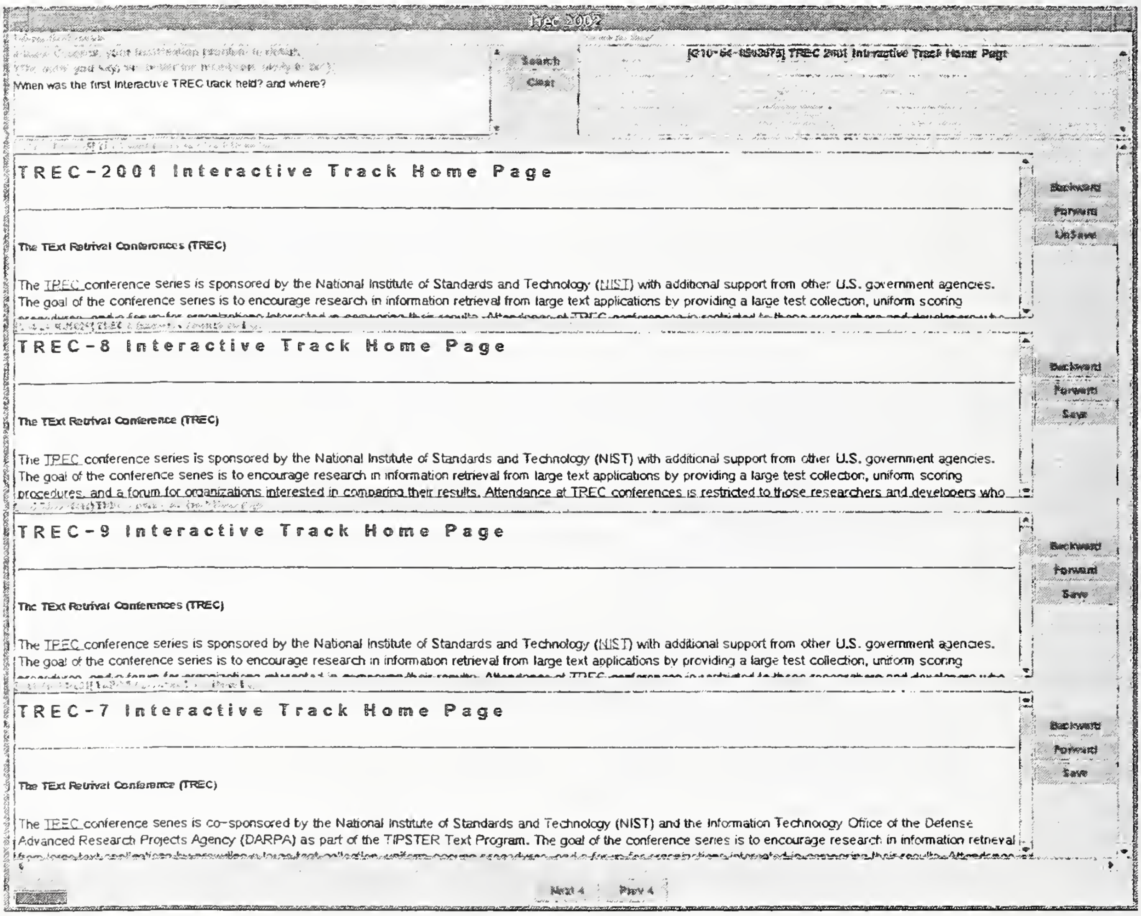
[1] http://www.excite.com
[2] http://trec.panopticsearch.com/

In addition, the actual implementations of the interfaces themselves may strongly influence user behavior. Therefore, we also present results with respect to usability of, and satisfaction with the interfaces and their various characteristics.

2 Systems, topics and database

In common with the other participants in the TREC 2002 Interactive Track, we used the Panoptic search engine, and the related TREC 2002 Web Track collection, as the basic retrieval system and database. We performed no modifications to the database or retrieval results. We also used the standard eight Interactive Track search topics to specify the tasks that our subjects would perform. Panoptic is basically a best-match search engine, but for queries of four words or less, it instead ranks documents according to a coordination-level algorithm. We decided that this difference would not affect hypothesis 2, since even for such queries, there should be a fairly close match to the results of the best-match algorithm.

All searches were performed using a Sun UltraSparc-IIi (440Mhz) with 512M memory and a 21 inch monitor. The two basic interfaces were implemented using Swing of Java 2 SDK, version 1.3. The four-document-at-a-time interface, called MDD, is shown in figure 1 (with the "information problem" query elicitation, called QE). The twenty-title interface, which used the standard Panoptic result format, called SDD, is shown in figure 2 (with the "query" query elicitation, c alled NQE)



When was the first interactive TREC track held? and where?

TREC-2001 Interactive Track Home Page

The TExt Retrival Conferences (TREC)

The TREC conference series is sponsored by the National Institute of Standards and Technology (NIST) with additional support from other U.S. government agencies. The goal of the conference series is to encourage research in information retrieval from large text applications by providing a large test collection, uniform scoring

TREC-8 Interactive Track Home Page

The TExt Retrival Conference (TREC)

The TREC conference series is sponsored by the National Institute of Standards and Technology (NIST) with additional support from other U.S. government agencies. The goal of the conference series is to encourage research in information retrieval from large text applications by providing a large test collection, uniform scoring procedures, and a forum for organizations interested in comparing their results. Attendance at TREC conferences is restricted to those researchers and developers who

TREC-9 Interactive Track Home Page

The TExt Retrival Conferences (TREC)

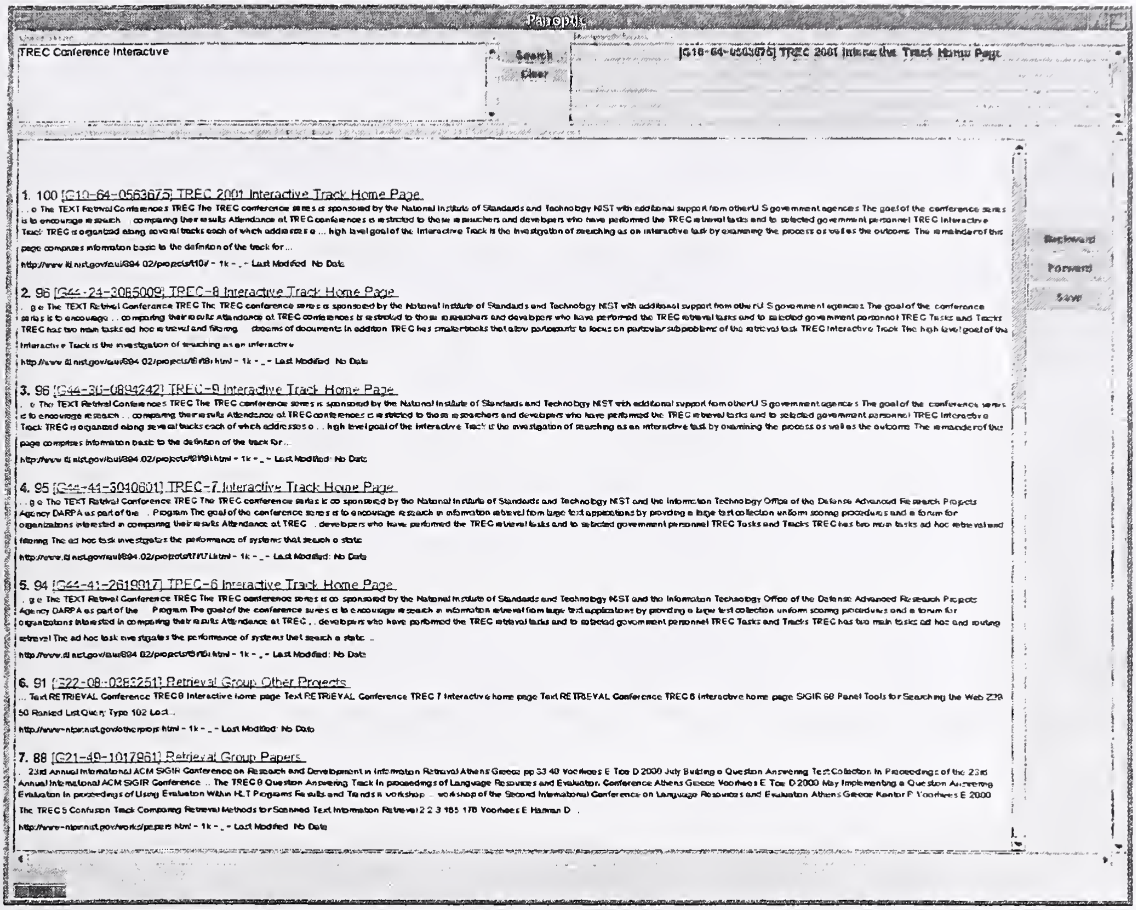
The TREC conference series is sponsored by the National Institute of Standards and Technology (NIST) with additional support from other U.S. government agencies. The goal of the conference series is to encourage research in information retrieval from large text applications by providing a large test collection, uniform scoring

TREC-7 Interactive Track Home Page

The TExt Retrival Conference (TREC)

The TREC conference series is co-sponsored by the National Institute of Standards and Technology (NIST) and the Information Technology Office of the Defense Advanced Research Projects Agency (DARPA) as part of the TIPSTER Text Program. The goal of the conference series is to encourage research in information retrieval

Figure 1. MDD interface, with query enhancement.



TREC Conference Interactive

[G10-64-0563675] TREC 2001 Interactive Track Home Page

1. 100 [G10-64-0563675] TREC 2001 Interactive Track Home Page
. . o The TEXT Retrieval Conferences TREC The TREC conference series is sponsored by the National Institute of Standards and Technology NIST with additional support from other U S government agencies The goal of the conference series is to encourage research . . comparing their results Attendance at TREC conferences is restricted to those researchers and developers who have performed the TREC retrieval tasks and to selected government personnel TREC Interactive Track TREC is organized along several tracks each of which addresses o … high level goal of the Interactive Track is the investigation of searching as an interactive task by examining the process as well as the outcome The remainder of this page comprises information basic to the definition of the track for …
http://www itl nist.gov/iaui/894 02/projects/t10i/ - 1k - _ - Last Modified No Date

2. 96 [G44-24-3065009] TREC-8 Interactive Track Home Page
. g e The TEXT Retrieval Conference TREC The TREC conference series is sponsored by the National Institute of Standards and Technology NIST with additional support from other U S government agencies The goal of the conference series is to encourage . . comparing their results Attendance at TREC conferences is restricted to those researchers and developers who have performed the TREC retrieval tasks and to selected government personnel TREC Tasks and Tracks TREC has two main tasks ad hoc retrieval and filtering streams of documents In addition TREC has smaller tracks that allow participants to focus on particular subproblems of the retrieval task TREC Interactive Track The high level goal of the Interactive Track is the investigation of searching as an interactive
http://www itl nist.gov/iaui/894 02/projects/t8i/t8i html - 1k - _ - Last Modified No Date

3. 96 [G44-30-0894242] TREC-9 Interactive Track Home Page
. e The TEXT Retrieval Conferences TREC The TREC conference series is sponsored by the National Institute of Standards and Technology NIST with additional support from other U S government agencies The goal of the conference series is to encourage research . . comparing their results Attendance at TREC conferences is restricted to those researchers and developers who have performed the TREC retrieval tasks and to selected government personnel TREC Interactive Track TREC is organized along several tracks each of which addresses o . . high level goal of the Interactive Track is the investigation of searching as an interactive task by examining the process as well as the outcome The remainder of this page comprises information basic to the definition of the track for …
http://www itl nist.gov/iaui/894 02/projects/t9i/t9i html - 1k - _ - Last Modified: No Date

4. 95 [G44-41-3040601] TREC-7 Interactive Track Home Page
. . g e The TEXT Retrieval Conference TREC The TREC conference series is co sponsored by the National Institute of Standards and Technology NIST and the Information Technology Office of the Defense Advanced Research Projects Agency DARPA as part of the . Program The goal of the conference series is to encourage research in information retrieval from large text applications by providing a large test collection uniform scoring procedures and a forum for organizations interested in comparing their results Attendance at TREC . developers who have performed the TREC retrieval tasks and to selected government personnel TREC Tasks and Tracks TREC has two main tasks ad hoc retrieval and filtering The ad hoc task investigates the performance of systems that search a static
http://www.itl nist.gov/iaui/894.02/projects/t7i/t7i.html - 1k - _ - Last Modified: No Date

5. 94 [G44-41-2619917] TREC-6 Interactive Track Home Page
. g e The TEXT Retrieval Conference TREC The TREC conference series is co sponsored by the National Institute of Standards and Technology NIST and the Information Technology Office of the Defense Advanced Research Projects Agency DARPA as part of the Program The goal of the conference series is to encourage research in information retrieval from large text applications by providing a large test collection uniform scoring procedures and a forum for organizations interested in comparing their results Attendance at TREC . . developers who have performed the TREC retrieval tasks and to selected government personnel TREC Tasks and Tracks TREC has two main tasks ad hoc and routing retrieval The ad hoc task investigates the performance of systems that search a static …
http://www.itl nist.gov/iaui/894 02/projects/t6i/t6i.html - 1k - _ - Last Modified: No Date

6. 91 [G22-08-0282251] Retrieval Group Other Projects
… Text RETRIEVAL Conference TREC 8 Interactive home page Text RETRIEVAL Conference TREC 7 Interactive home page Text RETRIEVAL Conference TREC 6 Interactive home page SIGIR 98 Panel Tools for Searching the Web Z39 50 Ranked List Query Type 102 Lost …
http://www-nlpir.nist.gov/otherprojs html - 1k - _ - Last Modified: No Date

7. 88 [G21-49-1017961] Retrieval Group Papers
. 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval Athens Greece pp 33 40 Voorhees E Tice D 2000 July Building a Question Answering Test Collection In Proceedings of the 23rd Annual International ACM SIGIR Conference . . The TREC 8 Question Answering Track In proceedings of Language Resources and Evaluation Conference Athens Greece Voorhees E Tice D 2000 May Implementing a Question Answering Evaluation In proceedings of Using Evaluation Within HLT Programs Results and Trends a workshop . workshop of the Second International Conference on Language Resources and Evaluation Athens Greece Kantor P Voorhees E 2000 The TREC 5 Confusion Track Comparing Retrieval Methods for Scanned Text Information Retrieval 2 2 3 165 176 Voorhees E Harman D .
http://www-nlpir.nist.gov/works/papers html - 1k - _ - Last Modified No Date

Figure 2. SDD interface, with no query enhancement

As can be seen from the screen shots, both interfaces had identical query entry boxes, and an identical list of saved documents. Saved documents in each interface could be opened for review, and unsaved if so desired. Each interface allowed subjects to follow links from displayed documents, whether the linked documents were in the Web Track collection, or on the live Web outside the collection. In the MDD system, subjects could page through the ranked document list four documents at a time; in the SDD system, subjects could page through the ranked title list twenty documents at a time. In general, seven to eight of the twenty titles were visible on the SDD screen without scrolling in the page; the first fifteen or so lines of a document were visible in each of the four MDD document panes, without scrolling. In MDD, documents could be saved directly by the appropriate button next to the displayed document; in SDD, they could be saved by following the link to the document, and then using the save button next to the saved documents list at the right top interface frame. Documents in SDD could also be saved directly from the results list, without following the link to the whole document, by selecting the relevant title and using the save button. However, this feature was not mentioned in the system demonstration, so it was used only rarely. When links within documents were followed, in either MDD or SDD, the searcher could return to the previous document by using the "Backward" button, and refollow links by using the "Forward" button. In SDD, returning to the search result list from a viewed page required using the "Backward" button.

3 Experiment design and conduct

We followed the basic Interactive Track within-subjects design for investigating the hypotheses related to interaction (1 & 3). With respect to the hypotheses related to query length (2& 4), we iterated the basic Interactive Track design twice, once in the QE condition, and once in the NQE condition, thus using a between subjects design. In both cases, subjects searched for answers to four topics using one interface, and then for four topics using the other interface. The assignment of subjects to conditions MDD and SDD, and the topics that were searched in each, is shown in table 1. This design was applied to the first sixteen subjects with query elicitation mode NQE, and repeated for the second set of sixteen subjects with query elicitation mode QE.

Subject	Block 1 System: Topics	Block 2 System: Topics
1	SDD: 4-7-5-8	MDD: 1-3-2-6
2	MDD: 3-5-7-1	SDD: 8-4-6-2
3	MDD: 1-3-4-6	SDD: 2-8-7-5
4	MDD: 5-2-6-3	SDD: 4-7-1-8
5	SDD: 7-6-2-4	MDD: 3-5-8-1
6	SDD: 8-4-3-2	MDD: 6-1-5-7
7	MDD: 6-1-8-7	SDD: 5-2-4-3
8	SDD: 2-8-1-5	MDD: 7-6-3-4
9	MDD: 4-7-5-8	SDD: 1-3-2-6
10	SDD: 3-5-7-1	MDD: 8-4-6-2
11	SDD: 1-3-4-6	MDD: 2-8-7-5
12	SDD: 5-2-6-3	MDD: 4-7-1-8
13	MDD: 7-6-2-4	SDD: 3-5-8-1
14	MDD: 8-4-3-2	SDD: 6-1-5-7
15	SDD: 6-1-8-7	MDD: 5-2-4-3
16	MDD: 2-8-1-5	SDD: 7-6-3-4

Table 1. Experimental design comparing MDD and SDD. NQE was used for the first 16 subjects, QE for the second set of 16 subjects.

All searching was done at the Information Interaction Laboratory at the School of Communication, Information and Library Studies (SCILS), Rutgers University. When subjects arrived, they were asked first to examine and sign the Informed Consent form[3]. They then completed a background questionnaire, eliciting various demographic data and data concerning searching experience. Next, the experimenter gave a demonstration of the first interface that the subjects would use, which was based on an example topic of the sort that the subjects would be searching on. The subjects were then given a paper form with a description of the first topic that they were to search on, and questions about whether they thought they knew the answer to the topic's question, and their confidence in that knowledge, which they answered at that time. Then, the subjects returned to the computer, were instructed that they would have up to ten minutes to complete the search, that they were to save those documents which helped them to answer the topic's question, and were asked to think aloud during the search. The computer monitor was videotaped during all searches, and the thinking aloud was recorded on the videotape. When the subjects thought they had answered the question, or when they had run out of time, the system was stopped, and the subjects were asked to fill out a questionnaire with respect to their satisfaction with the results of the search, and other characteristics of the search on that particular topic. This procedure was repeated for the next three topics. After the first four topics, subjects were asked to complete a questionnaire regarding their experience searching with that particular interface. They were then given a demonstration of the second interface that they were to use, and then the same procedure was followed for the next four topics. After the second post-system questionnaire, subjects were engaged in a semi-structured exit interview, which was tape recorded. This questionnaire elicited information about common features of the two interfaces, and also comparing the two interfaces. The entire procedure was typically finished in about two hours. All of the data collection instruments, and the scripts for the demonstrations, are available at http:/scils.Rutgers.edu/mongrel/trec2002/instruments

[3] Project approved by Rutgers IRB, number 01-407M.

4 Results

4.1 Subjects

Thirty-two volunteer subjects participated in this experiment. They were recruited largely from the student population at Rutgers SCILS (44% were full-time students), and some were given credit for participating in the experiment and writing a brief description of their experience. Twenty-six (81%) of the participants were female and 6 (19%) were male. Our subjects were most likely (47%) to be between 28-37 years of age, while their ages ranged overall from 18 to 57. Given our sampling strategy, it is unsurprising that the searchers in our study had a high level of education. Thirty-seven % had completed a Master's degree at the time of the experiment and nearly half (47%) said that they hoped to complete a Master's degree. Table 2 presents a descriptive profile of the searchers' level of experience with computers. It should be noted that all of the subjects were required to have some experience using Web search engines.

Experience:	N	Minimum	Maximum	Mean	Std. Deviation
Computers, general	32	4	7	6.28	.772
WWW browsers	32	5	7	6.38	.751
Computers at work	31	1	7	6.48	1.18
Academic computing	32	2	7	6.50	.984
Personal computing	32	2	7	6.66	.971
Entertainment	31	2	7	5.39	1.65
Search engines	32	5	7	6.28	.683
OPACS	32	3	7	5.44	1.16
Indexing Services	31	1	7	3.71	1.736

Table 2 Subject Experience with Computers (Based upon a 7 point scale in which 1= None 4=Some 7=A great deal)

Our subjects reported having an average of 6.2 years of searching experience. Using a 7 point scale to measure experience, in which 1=Novice and 7=Expert, the self-assessed level of expertise with computers was, on average, 5.19. Table 3, below presents the frequency with which the participants in our study engaged in a variety of searching activities. Two things are interesting to note from this table. First, our subjects engaged in these searching aetivities with a fairly high degree of frequency overall. Secondly, it is interesting that of all the searching activities we asked about, searching for government/policy information ranked last in terms of frequency, while searching for project related activities and for entertainment ranked highest.

Searching for:	N	Minimum	Maximum	Mean	Std. Deviation
Projects	32	2	7	5.84	1.05
Shopping	32	1	6	3.94	1.39
Traveling	32	1	6	3.53	1.52
Medical/health	32	1	6	3.34	1.66
Gov't/policy	32	1	6	2.56	1.48
Entertainment	32	1	7	4.44	1.52

Table 3 Subjects' Frequency of Searching (Based on a scale in which 1=Never 4=Monthly 7=Daily)

4.2 Measures and definitions

The variables used to characterize user searching behavior, and their definitions, are shown in table 4. Performance was measured by number of documents saved per search (cf. Belkin, et al., 2001), by user satisfaction with the search (on a seven-point scale, anchored by *Not at all* and *Extremely*, administered at the conclusion of each search), and by correctness and completeness of answer for the topic. Correctness and completeness were determined by comparing the pages which were saved for a search with judgments performed by experimenters at all of the TREC Interactive Track sites of all of the pages which were saved, at all sites, for each topic. Each page was judged as to whether it contained a correct answer to the topic, and if so, in cases where it was relevant, what aspects of the topic each page addressed. Thus, topics 1 ,2, 4, 5 and 6. which asked searchers to identify some specified number of pages or aspects could have incorrect, correct but incomplete, and correct and complete answers. Topics 3, 7 and 8, which asked for only one site or page, could have only correct or incorrect answers. In this paper, we consider an answer to be correct only if it is complete as well.

Variable	Definition
Pages seen	The total number of title references to pages displayed to the searcher through the course of the search (valid only for SDD)
Unique pages seen	The number of unique title references to pages displayed to the searcher (removing duplicate occurrences of references)
Pages viewed	The total number of pages whose contents were displayed to the searcher
Unique pages viewed	The number of unique pages whose contents were displayed to the searcher (removing duplicate occurrences of pages)
Number of documents saved	The total of all documents which were saved by the searcher through the course of the search
Number of final saved documents	The number of documents which were marked as saved at the conclusion of the search
Number of iterations	The total number of queries issued by the searcher, through the course of the search
Mean query length	The average length of all queries in a search, in words (both with and without stoplist applied)
Unique query length	The total number of unique words used in all of the queries in a search (both with and without stoplist applied)

Table 4. Variables used to describe search behavior

4.3 Descriptive statistics

Table 5 describes overall behaviors for all searches in both systems. The average number of the total pages seen and the average number of the unique pages seen were 145.16 and 56.38 respectively (relevant in SDD only). Meanwhile, the average number of the total pages viewed and the average number of the unique pages viewed were 13.64 and 10.60, in MDD and SDD together, respectively. On average, almost three documents (2.91) were ever saved by the subjects, and somewhat over 2 (2.33) were kept as finally saved documents. The subjects, on average, used just over two iterations (2.25) for their searching. Finally, subjects spent about 8 minutes and 21 seconds for each topic.

	Mean (Standard Deviation)	N
Total pages seen	145.16 (84.48)	128 (SDD only)
Unique pages seen	56.38 (39.05)	128 (SDD only)
Total pages viewed	13.64 (12.09)	255
Unique pages viewed	10.60 (9.68)	255
Documents ever saved	2.91 (2.18)	255
Final saved documents	2.33 (1.53)	255
Iterations	2.37 (1.52)	255
Time (seconds)	501.02 (195.06)	255

Table 5. Overall search characteristics, MDD and SDD together.

Search behavior ranged widely according to the topic (see table 6). First, the average total pages seen ranged from 116 to 185, and the average number of unique pages seen ranged from 37 to 84. All topics, except topic 2 (about 19 pages), had similar average total pages viewed, between 11 and 14. Also, the average numbers of unique pages viewed ranged from 9 to 11 except topic 2 (about 20 pages). Topic 7, with the smallest average number of final saved documents (1.66) had the largest average number of iterations (3.03) and unique pages seen (84.71). Conversely, topic 5, with the largest average number of final saved documents (2.97) showed the smallest average number of iterations (1.66) and unique pages seen (37.33). Subject used the least searching time for topic 5 (6 minutes and 33 seconds); the average was 8 minutes and 21 seconds.

	Total	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5	Topic 6	Topic 7	Topic 8
Total pages seen	145.16	161.00	134.67	116.82	150.40	116.00	147.06	150.59	185.40
Unique pages seen	56.38	63.53	56.00	44.94	49.33	37.33	43.53	84.71	70.20
Total pages viewed	13.64	12.53	19.53	12.38	10.81	12.88	13.84	13.97	13.13
Unique pages viewed	10.60	10.06	15.22	9.50	8.59	9.66	11.31	11.19	9.23
Number of document ever saved	2.91	3.03	2.56	2.87	3.03	3.28	3.69	1.78	3.00
Number of final saved documents	2.33	2.06	2.38	2.25	2.50	2.97	2.75	1.66	2.06
Number of iteration (queries)	2.37	2.28	2.78	2.03	1.66	1.66	2.13	3.03	2.45
Number of seconds taken	501.02	500.84	578.31	528.53	463.22	392.63	510.75	546.62	486.84

Table 6. Search characteristics by topic.

4.4 Interaction

Hypothesis 3 asked whether the MDD interface resulted in less user interaction than the SDD interface. Table 7 displays and compares the amount of interaction in each system, according to iterations, time (seconds), number of seen and viewed documents (total and unique), and ratios of unique to total seen and viewed, and in the case of SDD, ratios of seen to viewed, total and unique. For MDD, there were no seen documents, as the full texts of documents were always displayed to the subject. From table 7, we see that MDD had significantly more viewed documents than SDD, within a similar amount of time and number of iterations. Also, MDD subjects viewed far fewer documents than SDD subjects saw. While we did not log the amount of scrolling within particular documents or the number of times subjects paged to the next display of twenty titles in SDD or four documents in MDD, from the total number seen in SDD and total number viewed in MDD, we see that subjects paged less frequently in MDD (5) than in SDD (7). Although there was not a significant difference between the two systems in iterations or time, the differences are in the expected direction. On the basis of these data, we conclude that Hypothesis 3 is supported.

Hypothesis 1 asked whether a system designed to reduce the amount of interaction that a searcher has to engage in (i.e. make interaction more effective) will lead to increased satisfaction with the search results, and increased performance, as compared to a system not so designed. Table 8 displays and compares subjects' satisfaction with the search results and subjects' performance according to number of documents saved and number of correct answers. From table 8, we see that subjects were significantly more satisfied with their search results when searching with MDD than when searching with SDD. In terms of performance, subjects saved significantly more documents when searching with MDD than when searching with SDD, but the number of complete and correct answers to the topics did not vary significantly between interfaces.

Interaction Measure	MDD	SDD
Iterations	2.33 (1.52)	2.41 (1.53)
Time (seconds)	481.87 (199.74)	520.03 (189.05)
Number Seen (total)	N/A	145.16 (84.48)
Number Seen (unique)	N/A	56.38 (39.05)
Number Viewed (total)*	20.00 (13.76)	7.32 (4.90)
Number Viewed (unique)*	16.32 (10.73)	4.92 (2.83)
Ratio of unique seen to total seen	N/A	.44 (.24)
Ratio of unique viewed to total viewed*	.86 (.16)	.76 (.22)
Ratio of unique seen to unique viewed	N/A	.12 (.11)

Table 7. Interaction measures for MDD and SDD, mean and (standard deviation) (*p<.01)

Satisfaction or Performance Measure	MDD	SDD
Satisfaction with search results*	4.65 (2.00)	3.95 (2.09)
Number of documents saved*	2.77 (1.75)	1.91 (1.20)
Number of correct answers	93/127 = 73%	84/128 = 66%

Table 8. Satisfaction and Performance Measures for MDD and SDD, mean and (standard deviation) (*p<.01)

When combined with the interaction data above, the performance data provides additional evidence that MDD not only decreased interaction, but made interaction more effective . In the same number of iterations and in the same amount of time, subjects using MDD viewed significantly more documents and saved significantly more documents than those subjects using SDD. Subjects using MDD saved approximately 13% of the documents that they viewed, while subjects using SDD saved approximately 26% of the documents that they viewed, but only 1% of the documents that they saw. Of the documents that subjects using SDD saw, only 5% were viewed.

4.5 Query length

Hypothesis 4 asked whether the QE query elicitation mode resulted in longer query length than the NQE mode. Table 9 shows, for all searches in each condition, the mean query length, both with and without applying a stoplist. These figures are the mean of the number of words in each query in a search. The unique query length is the mean number of word types used in all queries in a search. Thus, the mean query length for a single search which used the two queries below is four (four terms in each query), while the unique query length is six (six unique words in the two queries).

Q1: usa congress privacy legislation Q2: usa congress electronic information

We interpret mean query length as a valid measure of the length of queries entered by the searcher. Unique query length, however, is interpreted as a measure of search effort, rather than of query length, since it measures the number of different words that the searcher had to think of over the course of the entire search.

	Mean iterations per search (SD)	Mean Query Length, stoplist (SD)	Mean Query Length, no stoplist (SD)	Unique Query Length, stoplist (SD)	Unique Query Length, no stoplist (SD)
NQE	2.64 (1.63)	4.24 (1.26)	4.85 (1.52)	5.97 (2.32)	6.85 (2.80)
QE	2.09 (1.35)	6.45 (3.00)	10.90 (7.30)	7.84 (3.34)	12.98 (7.33)

Table 9. Query statistics for NQE and QE modes, mean and (standard deviation).

Results from a t-test comparing QE and NQE on the basis of mean query length with stoplist indicate that searchers using the QE interface entered significantly longer queries (M=6.45; SD=3.00) than those using NQE interface (M=4.24; SD=1.26), t(253) = -7.67, p<.01. Thus, hypothesis 4 is strongly supported.

Hypothesis 2 asked whether a system which encouraged longer queries led to increased performance. Given the results with respect to hypothesis 4, we can investigate this hypothesis directly by comparing NQE with QE. As with interaction, we evaluate performance with three measures: searcher satisfaction; number of documents saved; and correctness of answer. There was no significant difference between NQE and QE in terms number of documents saved, or correctness of answer. However for satisfaction with search results, searchers were found to be more satisfied with their search results in QE (M=4.54; SD=1.96) than NQE (M=4.05; SD=2.15), although not quite significantly so, t(253) = -1.9, p=.058. So, we found only weak support for Hypothesis 2. Therefore, we investigated directly the relationship between query length and performance. In this analysis, significant correlations were found between satisfaction and mean query length, whether it is with (.137, p <.05) or without stop list (.136, p<.05). This seems to confirm a weaker version of hypothesis 2, that query length leads to better search outcome.

However, the data in table 9 show a negative significant relation between unique query length with stoplist and satisfaction (-.142, p <.05). This result is supported by analysis of correctness of response (table 10). Table 10 shows the relationship between correctness and unique query length, with stoplist and without. In both cases, more words in a search is significantly associated more strongly with incorrect answers than with correct answers ((253) = 2.78, p =.006; t(253) = 2.64, p=.009, respectively). These results support our interpretation of unique words in a search as a measure of search effort.

	correctness of a search	N	Mean (Standard Deviation)
Unique words in a search with stoplist	No	79	7.68 (3.13)
	Yes	176	6.56 (2.91)
Unique words in a search without stoplist	No	79	11.47 (7.15)
	Yes	176	9.23 (5.83)

Table 10. Unique words in query related to search correctness.

5 Discussion

We found support for hypothesis 3, that the MDD interface reduced interaction, and for hypothesis 1, that a system designed to make interaction more effective would lead to increased user satisfaction and increased performance. While subjects viewed significantly more full text documents in MDD than in SDD, they viewed significantly fewer documents in MDD than were seen in SDD. In terms of satisfaction and performance, subjects were significantly more satisfied in MDD than SDD, and saved significantly more documents in MDD than SDD. However, there was no difference in correctness of answers between the two treatments. Given that there were no differences in time and iterations between the two, these results indicate that because subjects were required to engage in more interaction in SDD than MDD, they had lower satisfaction, and decreased search effectiveness by one of two measures.

We found strong support for hypothesis 4, that the QE mode would lead to significantly longer queries than NQE. However, we found only weak support for hypothesis 2, that searchers in the QE mode would perform better than in the NQE mode. The somewhat weaker, related hypothesis, that longer queries would be associated with better performance, was only supported in part (with respect to mean query length being significantly associated with greater satisfaction with the search). However, in these circumstances, the number of iterations in a search might be considered an indirect measure of performance, if number of iterations is interpreted as effort needed to accomplish the task. The mean number of iterations per search (and standard deviation) for QE was 2.09 (1.35); for NQE, 2.64 (1.63). Results from a t-test indicate that subjects using QE had significantly fewer iterations than subjects using NQE, t(253) = 2.98, p<.01. Since there was no difference between correctness in the QE and NQE modes, we find further support for hypothesis 2, in that comparable results were achieved with less effort in QE than in NQE.

We found a significant negative relationship between unique query length and correctness of answer, as well as with satisfaction with a search. We speculate that these results might be explained by an interaction effect between unique query length and degree of interaction. As the number of iterations increases, the unique query length also increases. There is a strong correlation between iterations and unique query length (.44, p<.01). And the number of iterations is negatively correlated with satisfaction (-.53, p<.001). In other words, number of iterations might bc the common cause variable that leads to both longer unique query length and dissatisfaction. Therefore, when query length is averaged, instead of uniquely counted, the positive correlation between query length and satisfaction is revealed, because the iteration variable is held more-or-less constant. An alternative explanation is that both number of iterations and unique query length are indicators of difficulty of the search topic. These issues deserve further investigation.

6 Conclusions

Our results support the idea that reducing the amount of interaction required of a searcher, therefore making interaction more effective, leads to a better experience for searchers, and that the MDD interface, which displays documents directly for judgment and use, rather than requiring users to judge on the basis of a surrogate and then follow links to the documents, does make interaction more effective in just this way. If our speculations about interaction effects between iterations and query length are correct, such a result would tend to support the general interaction hypothesis. This leads us to conclude that alternatives to the web browser-based paradigm of displaying search results as lists of links which need to be traversed to get to actual documents need to be further investigated, and that displays which afford direct access to documents are likely to be preferable in several ways to lists of links.

We found also that query length in a Web searching environment can be substantially and significantly enhanced by using a rather simple interface technique. Enhancing queries length in this way led to some increase in users' satisfaction with search results, and to significant increase in effectiveness of searches, c onsidered as degree of effort required to achieve specific level of performance in the search task. Thus, we see support both for the possibility of increasing search length in interactive IR, and for the utility of doing so, at least in best-match search systems. These results suggest that IR systems need not be bound by the finding that queries presented to current systems are short, especially since most interfaces in current systems are designed to elicit short queries. In particular, the results suggest that much more thought should be given to how to elicit information problem descriptions in interactive IR systems. And, they suggest an alternative to pseudo-relevance feedback and similar techniques for enhancing query length, that may be more closely related to searcher needs than those techniques.

7 Acknowledgements

We wish to thank our colleagues who worked so hard with us in designing, scheduling and running the experiments: Amymarie Keller, Yuelin Li and William Voon, and our wonderful volunteer subjects. The work reported in this paper was supported in part by NSF Grant Number IIS 9911942. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation (NSF).

8 References

Belkin, N.J., Keller, A., Kelly, D., Perez-Carballo, J., Sikora, C., Sun, Y. (2001) Support for Question-Answering in Interactive Information Retrieval: Rutgers' TREC-9 Interactive Track Experience. In E.M. Voorhees and D.K. Harman (eds.) *The Ninth Text Retrieval Conference, TREC 9*. (463-475).

Belkin, N.J., Jeng, J., Keller, A., Kelly, D., Kim, J., Lee, H.-J., Tang, M.-C., Yuan, X.-J. (2002) Rutgers' TREC 2001 Interactive Track Experience. In E.M. Voorhees and D.K. Harman (eds.). *The Tenth Text Retrieval Conference, TREC 2001* (465-472). Washington, D.C.: GPO.

Rutgers Filtering Work at TREC 2002: Adaptive and Batch

Andrei Anghelescu[d], Endre Boros[i], David Lewis[m], Vladimir Menkov[a], David Neu[c] and Paul Kantor[s]
angheles@cs.rutgers.edu, boros@rutcor.rutgers.edu, ddlewis@worldnet.att.net
vmenkov@aplab.rutgers.edu, djneu@acm.org, kantor@scils.rutgers.edu *

ABSTRACT

This year at TREC 2002 we participated in the adaptive filtering sub-task of the filtering track with some models for training a Rocchio classifier. Results were poorer than average on the utility type measures. Using simple feature selection produced better than average results on an F-type measure. The key to our approach was the use of pseudo-judgments, and an approach to threshold updating. We also participated in the batch filtering sub-task of the filtering track and investigated the use of rank based feature selection techniques in conjunction with a very simple classification rule.

1. INTRODUCTION

In the adaptive filtering sub-task of the filtering track, systems utilize a training set consisting of a small set of documents which are labelled either *relevant* or *irrelevant* . This is supplemented by a training set, from which one may draw inferences about the corpus, and may hazard some conjectures as to the relevant documents. In the work reported here, a simple version of "pseudo-relevance feedback" is used to expand the terms appearing in the 3 relevant documents, and the original topic statement.

Our approach in preparing to study the problem of adaptive filtering attempts to:

–Incorporate major techniques common to high-scoring AF approaches in recent TRECs.

–Allow easy modification of aspects we are likely to be doing experiments on.

–Be efficient enough to do many tuning runs.

–Be as simple as possible to implement given the above constraints.

We prepared the AP 1988-1990 data, which served as a "sandbox" for the selection of parameters in the adaptive Rocchio model. We used the LEMUR [10] toolkit to manage the text, build indices, etc.

We introduced a number of parameters controlling how many examples will be pseudo-labeled and with what weights:

In the batch filtering sub-task of the filtering track, systems utilize a training set consisting of documents which are labelled either *relevant* or *irrelevant* for a given information needs to develop static classifiers which attempt to distinguish the documents labelled relevant from those labelled irrelevant. In our opinion, efforts to attack this problem are often complicated by several characteristics of textual data.

Textual data is generally represented by using the terms in the text as features. Such data is inherently *highly dimensional* — the number of features being potentially equal to the number of words in the English language. In addition, misspellings, the improper, or colloquial use of words, and the fact that many very common words (e.g. "a", "and", "the", etc.) are virtually useless for distinguishing relevant documents from irrelevant ones, regardless of the information need, lead textual data to be *noisy*. Finally, most terms, even those not considered "noise" under the previous description, are not needed to distinguish relevant documents from irrelevant documents.

The aforementioned characteristics of textual data indicate that it might be possible to represent a document collection using only a subset of the original feature set which is much smaller than the original feature set, yet possesses properties which serve to facilitate the process of distinguishing relevant documents from irrelevant documents.

The idea we pursued in the batch filtering sub-task of the filtering track was to employ a heuristic designed to generate such feature subset and to then train an extremely simple classifier on the training set represented only in terms of the selected feature set.

2. ADAPTIVE FILTERING: BUILDING A CLASSIFIER

*Research supported in Part by the National Science Foundation under Grant NumberEIA-0087022. PBK and KBN are supported in Part by Advanced Research and Development Activity (ARDA)'s Advanced Question Answering for Intelligence (AQUAINT) Program under contract number 2002-H790400-000, the HITIQA project, of SUNY Albany and Rutgers. EB is supported in part by the Office of Naval Research (Grant N00014-92-J-1375). The views expressed in this article are those of the authors, and do not necessarily represent the views of the sponsoring agency. Author Affiliations: $^d =^c$ Division of Computer Sciences, Rutgers, the State University of New Jersey; ^{i} Rutgers Center for Operations Research; (^{m}). Independent Consultant, Chicago IL. ^{a}. Independent Consultant, Penticton, British Columbia Canada; ^{s} SCILS and DIMACS, Rutgers, the State University of New Jersey

2.1 Initialization

Initial training for each topic uses the training set (on which relevance status with respect to the topic is known only for three of the documents), and the topic description.

FOR EACH Topic

B1. Read topic description

B2. Read initial 3 positive training examples for this topic. Give each of these examples a weight of 1.0.

B3. Call scoring model learning algorithm (Rocchio) to produce linear model based on the topic description and the initial positive examples.

B4. Call the "pseudolabeling algorithm" to run the linear model trained in Step B3 against *all* training documents. It will return some portion of the training documents, and will have associated with them positive or negative pseudolabels, and fractional weights. Essentially, the documents that achieve a high score on vector retrieval with the initial query and 3 positive documents are taken as "relevant", those with low score are taken as "not relevant". Our algorithm actually has a number of parameters that control (a) the dividing line between "pseudo-relevant" and "pseudo-irrelevant" documents (which we refer to, together, as the "pseudo-labelled" documents) (b) the fractions of each class that are sampled into the updated Rocchio classifier and (c) the weights that each type of "pseudo" document are assigned in step B5. Eventually these weights are expressed in terms of the number of "equivalent documents" that the pseudo-labelled documents represent.

B5. Call the classifier learning algorithm, which changes the query, *a la* Rocchio, and selects a threshold that maximizes the target score, on the training set.

Numerous implementation details are not described here. Note that the idea of an "outer loop" over topics represents just one way to approach the problem, which may not be optimal for specific choices of the learning algorithm.

For further analytical work we have since modified the code to save the classifiers or the internal state of the training algorithm to persistent storage after initial training. This will potentially be useful for multiple experiments with the same starting point, as well as for comparative experiments (studying improvement of classifier over time).

2.2 Adaptive Phase of Training

In adaptive filtering, we run through the test documents in the specified order, applying classifiers, getting judgments only for documents judged relevant, and updating the classifiers.

FOR EACH Topic

FOR EACH Test Document

C1. Apply current classifier for topic to test document, computing score and determining if score is above threshold.

IF score is ≥ threshold

C2.1. Pass document ID, topic ID, score, and label ("relevant") to routine that writes output for evaluation

C2.2. Pass document ID and topic ID to judging routine, which will return label (Relevant vs. Nonrelevant vs. Unjudged).

ELSE

C3.1 Label = Unknown

C4. Pass current classifier, document ID, Label, and a weight of 1.0 to Learner (which for the baseline will be an object that in turn calls Rocchio and TROT).

2.3 the Rocchio Algorithm

The Rocchio algorithm [9] produces a linear model, which must then be specified with a threshold. The basic inputs to the algorithm are:

1. An initial "query" vector
2. A set of document vectors. Each vector is accompanied by a weight and a label.
3. The Rocchio weighting parameters (α, β, γ)
4. Feature weighting parameters
5. Feature selection parameters and rules

The Rocchio algorithm [9, 4] is a batch algorithm. It produces a new weight vector w from an existing weight vector w_1 and a set of training examples. The jth component w_j of the new weight vector is:

$$w_j = \alpha w_{1,j} + \beta \frac{\sum_{i \in C} x_{i,j}}{n_C} - \gamma \frac{\sum_{i \notin C} x_{i,j}}{n - n_C} \qquad (1)$$

where n is the number of training examples, $C = \{1 \leq i \leq n : y_i = 1\}$ is the set of positive training examples (i.e., members of the class of interest), and n_C is the number of positive training examples. The parameters α, β, and γ control the relative impact of the original weight vector, the positive examples, and the negative examples, respectively.

Typically, classifiers produced with the Rocchio algorithm are restricted to having nonnegative weights, so that instead of using the raw w from Equation (1), one uses w' where

$$w' = \begin{cases} w & \text{if } w > 0 \\ 0 & \text{otherwise.} \end{cases}$$

This is turned into a classifier by the relatively expensive process of recomputing the threshold after each new judgment is received on a submitted document. The computation of the threshold can be somewhat accelerated with a full Rocchio model, but we have not found a way to accelerate it meaningfully when a non-linear step such as the selection of a number of "top features" is included.

2.4 Retaining only the top 30 terms in a query

To improve performance, we limited the number of terms appearing in a query.

The specific algorithm is given in pseudocode as

Algorithm 1: Query term selection

```
Require: query vector $Q$, k
for $t \in Q$ do
    if $t < 0$ then
        $t = 0$
    end if
end for
$S = reverse(sort(Q))$
Ensure: $S[1 : min(|S|, k)]$, the top $k$ positive components of $Q$
```

3. ADAPTIVE TRAINING HEURISTICS

To find a Rocchio classifier we started at "plausible" values for all of the parameters in the model, and conducted a "greedy" search on each of the parameter values separately.

Original results concentrated on the utility based measures, and were terrible. This led to the development of a "TREC-specific" feature, which stops sending examples for judgment if the rate of success falls too low. One such heuristic is to stop when the number of consecutive negatives exceeds the total accumulated positive judgments obtained. Such heuristics have no meaning in the real world situations to which adaptive filtering will be applied.

An alternative heuristic, which can be justified for real applications, is to reduce the number of components in the updated Rocchio vector to a very small number. In one such run the number of components is reduced to 30. These 30 components are selected on the basis of their individual explanatory power, with regard to the specific measure of performance under considerations. In the submitted run, this was an F-measure.

Since F measures can be rewritten as $\frac{1}{\beta\frac{1}{p}+(1-\beta)\frac{1}{R}}$ they are very sensitive to finding *any* relevant documents. If (g, G) are the numbers of relevant documents (found, in the collection) respectively, and n documents are returned,

$$F = 1/(\beta n/g + (1-\beta)G/g) = g/(\beta n + (1-\beta)G)$$

.

So a system that "hangs in there" and eventually produces even a single relevant document will score better than a more discriminating system that returns no relevant documents, and quits sooner.

The results of our early experiments show only that we have set up a workable laboratory for exploring a host of possible combinations of the five key ingredients of an adaptive algorithm: these ingredients are a compression rule; a representation rule; a matching scheme, a learning scheme, and a fusion or selection scheme for combining multiple approaches to each of these five components. As is well known in the information retrieval community, the adaptive filtering task is extremely difficult, but we are optimistic that previously unexplored combinations of approaches may yield meaningful improvements in performance. The results are showin in Table 2, which appears at the end of the paper. The meaning of the row and column labels is as follows.

1. label of the run, which is composed of 3 parts - the value of the weight of the unjudged documents (parameter thres.unjWt - U-xx → thres.unjWt=xx) followed by the name of the parameter that is changed and the utility that is optimised (for example "best.f" means that the f-beta utility is optimised)

   The parameter related labels have the following meanings:

   - A+ : $\alpha = 2.0$
   - A- : $\alpha = 0.5$
   - C+ : $\gamma = 0.25$
   - C- : $gamma = 0.0625$
   - ND+ : neg density s.t. 2000 pseudo negatives are selected
   - ND- : neg density s.t. 500 pseudo-negatives are selected
   - PD- : pos density = 0.5, corresponding to 10 pseudo-positive documents
   - PW+ : pos weight of 5
   - PW- : pos weight of 1
   - NW+ : neg weight of 10
   - NW- : neg weight of 2
   - def : default values, $\alpha = 1.0, \beta = 1.0, \gamma = \frac{1}{8}$, ND s.t. 1000 pseudo-negative documents are selected, PD s.t. 20 pseudo-positive documents are selected, PW = 2, NW=5%

2. the number of topics that obtained a positive score in the test
3. min score - the lowest topic score
4. the total score (sum of all topic scores)
5. average T11U score
6. average T11F score
7. average T11SU score
8. the number of topics in this run that found at least 1 positive doc
9. the number of topics in this run that found at least 3 positive docs
10. the number of topics for which at least one document was sent to the
11. the "giveup threshold" for which these results were obtained oracle.

3.1 Ratio Based Scoring

In order to provide variety, we also used an alternate scoring scheming in which documents are ordered by a measure of the ratio of their similarities to the centroids of the positive and negative examples. Thus it builds on the relevance feedback information available to Rocchio, with a key difference. Scores are calculated using the (regularized) ratio of distances between normalized vectors. Specifically, if p, n are the unit vectors corresponding to the centroids of the positive and negative examples, and d is the unit vector corresponding to the document being scored, then

$$s_C(d) = \frac{1-(\mathbf{n},\mathbf{d})}{1-(\mathbf{p},\mathbf{d})} \qquad (2)$$

If the denominator vanishes, the value 10^6 is used as a default.

In practice this was more effective with a "Quitting" rule that cust off submission if, after the first 50 documents are submitted, we have not achieved a postive utility score.

4. BATCH FILTERING: BOOLEAN MODEL

Assume that there are $n > 0$ distinct terms in the document collection and associate an index in $V = \{1, 2, \ldots, n\}$ with each of these terms. Letting $\mathbb{B} = \{0, 1\}$, we represent each document in the collection as an n-dimensional Boolean vector $\mathbf{x} \in \mathbb{B}^V$. Each component of $\mathbf{x}$ corresponds one of the distinct terms in the document collection, with $x_i = 1$ if the i^{th} term is present in the document and $x_i = 0$ if the i^{th} term is absent from the document.

For a subset $S \subseteq V$, and vector $a \in \mathbb{B}^V$, we shall let $a[S] \in \mathbb{B}^S$ denote the projection of a onto S and for $X \subseteq \mathbb{B}^V$ we shall write $X[S]$ as the projection of X on S, that is,

$X[S] = \{a[S] \mid a \in X\}$. For a subset $S \subseteq V$ let us denote by $\chi^S \in \mathbb{B}^n$ its *characteristic vector*, i.e.

$$\chi_j^S = \begin{cases} 1 & \text{if } j \in S, \\ 0 & \text{otherwise.} \end{cases}$$

We shall refer to the set of relevant documents as T and the set of irrelevant documents as F and shall assume that $T \cap F = \varnothing$, that is, there do not exist vectors $a \in T$ and $b \in F$ such that $a = b$.

A set $S \subseteq V$ is said to be a *support set* for T and F if it has the property that $T[S] \cap F[S] = \varnothing$. That is, S is a support set if each relevant document represented in terms of the selected features subset can be distinguished from each irrelevant document represented in terms of the selected features subset.

The document model described above does not preserve information about the order in which terms appear in the document and therefore is often referred to as the *bag-of-words* representation. In addition, the Boolean nature of this representation lies in contrast to a popular representation known in the information retrieval literature as the *vector space model*, in which the components x_i correspond to the (relative) frequency of the term in the document.

4.1 Measure of Separation

For a subset $S \subseteq V$, we measure the distance between the projections $T[S]$ and $F[S]$ of the sets T and $F \in \mathbb{B}^V$ onto $\mathbb{B}^S$, by the so called *average Hamming distance*. The use of Hamming distance based separation, rather than measures based on the l_1, l_2 or l_∞ norms, as is often the practice when the employing the vector space model, is suggested by the Boolean nature of our document model.

The *Hamming distance* between the vectors $a[S] \in T[S]$ and $b[S] \in F[S]$ is defined as $d_S(a, b) = \sum_{j \in S: a_j \neq b_j} 1$. The average Hamming distance between the sets $T[S]$ and $F[S]$ then is defined as

$$\Delta_{avg}(S) = \frac{1}{|T||F|} \sum_{a \in T} \sum_{b \in F} d_S(a, b). \qquad (3)$$

4.2 Ranking Functions

For each $i \in V$, each of the ranking functions presented here utilizes the following four values

- $a_i \equiv$ the number of relevant documents containing the i^{th} term
- $b_i \equiv$ the number of irrelevant documents containing the i^{th} term
- $c_i \equiv$ the number of relevant documents which do not contain the i^{th} term
- $d_i \equiv$ the number of irrelevant documents which do not contain the i^{th} term

For each $i \in V$, the relationship between a_i b_i, c_i and d_i and the document collection is given by the following 2×2 contingency table

	$y \in T$	$y \in F$	
$x_i = 1$	a_i	b_i	$a_i + b_i = \theta_i$
$x_i = 0$	c_i	d_i	$c_i + d_i = \bar{\theta}_i$
	$a_i + c_i = \lvert T \rvert$	$b_i + d_i = \lvert F \rvert$	m

where the marginals θ and $\bar{\theta}_i$ represent the number of documents containing the i^{th} term and the number of documents which do not contain the i^{th} term respectively, and $y \in \mathbb{B}$ is defined as

$$y = \begin{cases} 1 & \text{if } x \in T, \\ 0 & \text{otherwise.} \end{cases}$$

Obviously, the marginals $|T|$ and $|F|$ are constant for all terms while the marginals θ_i and $\bar{\theta}_i$ vary for each term. The total number of documents in the collection is $m = a_i + b_i + c_i + d_i$ which is obviously also a constant.

For the simplicity of notations, we shall view all ranking functions as functions of the four parameters a, b, c and d, though clearly there are only two independent values among these.

In [2] we analyzed and compared a number of possible ranking functions, and based on that study, we selected 5 such functions for this TREC experiment:

Function α

$$\alpha = \left| \frac{a}{a+c} - \frac{b}{b+d} \right| = \frac{|ad - bc|}{|T||F|} \qquad (4)$$

is the absolute value of the difference between the number of relevant-irrelevant document pairs in the training collection which provide evidence that the i^{th} term is a good classifier of relevant documents and the number of relevant-irrelevant document pairs which provide evidence that the i^{th} term is a good classifier of irrelevant documents, normalized by the total number (i.e. both correctly distinguished and incorrectly distinguished) of relevant-irrelevant document pairs.

Function β

$$\beta = \frac{ad + bc}{(a+c)(b+d)} = \frac{ad + bc}{ab + ad + bc + cd} = \frac{ad + bc}{|T||F|} \qquad (5)$$

is the total number of relevant–irrelevant document pairs correctly distinguished by the i^{th} term, normalized by the total number of relevant-irrelevant document pairs in the training collection.

Function γ

$$\gamma = \frac{ad}{(a+c)(b+d)} = \frac{ad}{|T||F|}$$

is an obvious variant of both α and *beta*.

Function δ

$$\delta = \frac{|ad - bc|}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} = \frac{ad - bc}{\sqrt{\theta\bar{\theta}|T||F|}} \qquad (6)$$

is the absolute value of the *Pearson Product Moment Correlation* coefficient or simply the *correlation coefficient* for the Boolean variables x_i and y as defined above. It measures the degree to which these two variables have a linear relationship.

Function ρ

$$\rho = \frac{(a+b+c+d)\,(ad - bc)^2}{(a+b)(c+d)(a+c)(b+d)} = \frac{m(ad - bc)^2}{\theta\bar{\theta}|T||F|} \qquad (7)$$

is the χ^2 statistic for the Boolean variables x_i and y as defined above and provides another measure of association for these two variables.

Note that ρ is a onotone funcion of δ, so that our precedure. as described below. effectively gives a "double weight" to this particular measure of effectiveness.

4.3 Training the Batch Classifier

This section describes the feature selection method and the simple classifier used in the batch filtering sub-task of the filtering track.

The set of unique terms in the training set $T \cup F$ was ranked by each of the five ranking functions $\alpha, \beta, \gamma, \delta, \rho$ described in §4.2. Five intermediate feature sets, S_α, S_β, S_γ, S_δ, S_ρ, were constructed using the top ranking $K = 50$ terms of the corresponding ranking functions. Letting

$$\tilde{S} = S_\alpha \cup S_\beta \cup S_\gamma \cup S_\delta \cup S_\rho$$

we assigned a score $\psi \in \{1, \cdots, 5\}$ to each of the terms in *tiledS*, defined as the number of sets S_ξ, $\xi \in \{\alpha, \beta, \gamma, \delta, \rho\}$ in which the term appeared. The final feature set S was constructed by selecting the $K = 50$ terms with the highest ψ scores.

Next, to each term in $i \in S$ we assigned the weight

$$\omega(i) = \frac{\frac{a_i+0.5}{a_i+c_i+1}}{\frac{b_i+0.5}{b_i+d_i+1}}$$

which can be seen to be the Bayesian weight of evidence, and to each document $y \in T[S] \cup F[S]$ we assigned the score

$$\Omega(y) = \sum_{j \in S} \log(\omega(i)) y_j.$$

That is, each document projected onto the selected feature set S is assigned a score equal to the sum of the logarithms of the Bayesian weights of evidence for the terms it contains.

The batch filtering task requires the definition of a static classification rule which specifies whether each document in the test set should be considered relevant and retrieved, or irrelevant and ignored. The rule we utilized specifies that $y \in T[S] \cup F[S]$ will be retrieved if and only if $\Omega(y) \geq \tau$ for some $\tau \in \mathbb{R}$. The threshold τ was selected so as to optimize the utility measure $TU11 = 2R - I$ over the training set, where R is the number of relevant documents retrieved by the system and I is the number of irrelevant documents retrieved by the system.

5. RESULTS

5.1 Filtering Results

Our training results, using a variety scoring measures, for a great variety of training runs, are shown at the end of the paper in Table 2. In the final analysis, our results at TREC were in the middle of the pack.

These are summarized in Table 1.

Method	Mean T11
dimacsddl02a	0.110
dimacs11aAPQ	0.142
dimacsddl02b	0.293
dimacs11aP1Q	0.272
dimacs11a30Q	0.337

Table 1: TREC 2002 Results for the Assessor topics, various runs

The best results were achieved by the run submitted as *dimacs11a30Q*. This was a Rocchio method, trained on a set of documents similar to the one used at TREC. The " 30" indicates that only the top 30 terms, that is, the 30 terms with highest weight in the updated query vector were included. " AP" includes only the terms with positive weight are retained. " Q" indicates that for our final submission we cut off submission if we did not achieve a positive score after submitting 50 documents for judgment. " P1Q" used a ratio scoring scheme, together with the " quit at 50 if score is negative rule. This is, of course, a " TREC strategy" and not a procedure that would be useful in a real world application.

We have subsequently learned that with proper learning parameters, as chosen by the group from the Chinese Academy of Sciences, it is possible for a Rocchio approach similar to ours to achieve very good results. We are not certain as to which steps of our approch blocked us from realizing this high level of performance. One possibility is that even the small number of pseudo-negative cases that we introduce into the training is sufficient to keep us away from the region of good performance. Another is that the space of parameters is too large, and the dependence of the learning too complex, to be successfully explored "one variable at a time", which was essentially the heuristic used. Other inhibiting factors may have included the heursitcs used to cut off submission if we did not achieve a positive score after the first 50 judgments. Nonetheless, our submission that *did* use this heuristic fared better than those that did not.

5.2 Batch Results

On the *assessor judged topics* our TU11 score was less than the median fifteen times, equal to the median twenty times, and greater than the median fifteen times and never attained the maximum.

On the *intersection topics* our TU11 score was less than the median once, equal to the median six times, and greater than the median forty-three times and attained the the maximum twenty-one times. Unfortunately, for many of these topics, submitting no documents at all was an effective TREC strategy.

6. CONCLUSIONS

This work is part of a larger effort to develop an array of approaches to filtering problems, and to integrate or fuse them for greater effectiveness. In this first effort it would appear that we have adopted tools that are capable of "state of the art" perfromance on the adaptive filtering task, but have not yet learned how to ensure that this level of performance is achieved.

7. ACKNOWLEDGMENTS

This research is supported in part by the National Science Foundation, which is not, however, responsible for any positions expressed in this report. We thank our colleagues in the Monitoring Message Streams project: Fred Roberts (Principal Investigator), David Madigan, Ilya Muchnik, S. Muthukrishnan, Rafi Ostrovsky, and Martin Strauss for helpful conversations.

8. REFERENCES

[1] Endre Boros, Takashi Horiyama, Toshihide Ibaraki, Kazuhisa Makino, and Mutsunori Yagiura. Finding essential attributes from binary data. *Annals of Mathematics and Artificial Intelligence*, accepted.

[2] Endre Boros and David J. Neu. Rank based feature selection in information retrieval. Technical report, RUTCOR, Rutgers University, 2002.

[3] William B. Frakes and Ricardo Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms.* Prentice-Hall PTR, 1992.

[4] Donna Harman. Relevance feedback and other query modification techniques. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 241–263. Prentice Hall, Englewood Cliffs, NJ, 1992.

[5] D. D. Lewis. Feature Selection and Feature Extraction for Text Categorization. In *Proceedings of Speech and Natural Language Workshop*, pages 212–217, San Mateo, California, 1992. Morgan Kaufmann.

[6] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[7] Gottfried E. Noether. *Introduction to Statistics: The Nonparametric Way.* Springer-Verlag, 1991.

[8] S.E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[9] J. J. Rocchio, Jr. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971.

[10] Chengxiang Zhai, Thi Nhu Truong, John Lafferty, Jamie Callan, David Fisher, Fangfang Feng, James Allan, and Bruce Croft. *Lemur.* http://www-2.cs.cmu.edu/ lemur, 2001.

Test label	2	3	4	5	6	7	8	9	10	11
U-0.0_A+	9	-83	67	1.340	0.166	0.266	27	24	50	n/a
U-0.0_A-	8	-234	-529	-10.796	0.158	0.250	31	23	49	n/a
U-0.0_C+	12	-67	109	2.180	0.195	0.274	35	30	50	n/a
U-0.0_C-	13	-86	-79	-1.580	0.175	0.268	30	23	50	n/a
U-0.0_ND+	14	-77	143	2.860	0.191	0.278	33	25	50	n/a
U-0.0_ND-	14	-77	120	2.400	0.190	0.275	34	25	50	n/a
U-0.0_NW+	13	-80	53	1.060	0.179	0.282	30	24	50	n/a
U-0.0_NW-	11	-109	-142	-2.840	0.173	0.269	31	25	50	n/a
U-0.0_PD-	11	-61	49	1.043	0.168	0.273	27	20	47	n/a
U-0.0_PW+	8	-374	-703	-14.060	0.165	0.224	33	26	50	n/a
U-0.0_def	16	-51	325	6.771	0.191	0.287	32	23	48	n/a
U-0.1_A+.best.f	15	-34	499	9.9800	0.2040	0.3030	34	26	50	184
U-0.1_A-.best.f	15	-63	103	2.1460	0.1730	0.2690	30	22	48	293
U-0.1_C+.best.f	17	-44	347	7.2290	0.2110	0.2880	36	30	48	161
U-0.1_C-.best.f	11	-54	260	5.4170	0.1750	0.2850	28	20	48	186
U-0.1_ND+.best.f	17	-50	405	8.4380	0.1940	0.2930	32	23	48	211
U-0.1_ND-.best.f	16	-50	370	7.7080	0.1930	0.2900	32	23	48	212
U-0.1_NW+.best.f	12	-50	338	7.0420	0.1870	0.2890	30	22	48	249
U-0.1_NW-.best.f	13	-50	317	6.6040	0.1900	0.2880	32	24	48	192
U-0.1_PD-.best.f	13	-56	242	5.1490	0.1630	0.2960	27	18	47	147
U-0.1_PW+.best.f	14	-65	181	3.6200	0.1960	0.2800	34	25	50	217
U-0.1_PW-.best.f	15	-30	331	6.8960	0.1820	0.2900	30	24	48	191
U-0.1_def.best.f	16	-50	368	7.6670	0.1930	0.2900	32	23	48	211
U-0.25_A+.best.f	16	-34	567	11.3400	0.2100	0.3060	34	26	50	176
U-0.25_A-.best.f	16	-63	144	3.0000	0.1780	0.2740	30	22	48	235
U-0.25_C+.best.f	16	-44	381	7.9380	0.2100	0.2910	35	30	48	157
U-0.25_C-.best.f	12	-53	278	5.7920	0.1750	0.2860	28	20	48	184
U-0.25_ND+.best.f	17	-49	443	9.2290	0.1960	0.2950	32	24	48	244
U-0.25_ND-.best.f	16	-49	395	8.2290	0.1950	0.2920	32	24	48	216
U-0.25_NW+.best.f	12	-50	353	7.3540	0.1880	0.2900	30	22	48	246
U-0.25_NW-.best.f	12	-48	313	6.5210	0.1920	0.2900	32	25	48	335
U-0.25_PD-.best.f	13	-55	248	5.2770	0.1630	0.2960	27	18	47	146
U-0.25_PW+.best.f	14	-70	141	2.8200	0.1930	0.2810	34	24	50	188
U-0.25_PW-.best.f	15	-30	356	7.4170	0.1820	0.2910	30	23	48	194
U-0.25_def.best.f	16	-49	394	8.2080	0.1940	0.2920	32	24	48	210
U-0.5_A+.best.f	16	-33	566	11.3200	0.2100	0.3070	34	25	50	176
U-0.5_A-.best.f	15	-52	185	3.8540	0.1800	0.2760	30	23	48	231
U-0.5_C+.best.f	16	-41	390	8.1250	0.2060	0.2920	34	28	48	313
U-0.5_C-.best.f	12	-35	306	6.3750	0.1730	0.2880	27	20	48	185
U-0.5_ND+.best.f	17	-48	468	9.7500	0.1950	0.2960	32	24	48	223
U-0.5_ND-.best.f	16	-48	418	8.7080	0.1940	0.2930	32	24	48	223
U-0.5_NW+.best.f	12	-49	369	7.6880	0.1870	0.2920	29	22	48	252
U-0.5_NW-.best.f	13	-47	291	6.0620	0.1890	0.2860	32	25	48	337
U-0.5_PD-.best.f	13	-54	258	5.4890	0.1640	0.2970	27	18	47	143
U-0.5_PW+.best.f	13	-68	121	2.4200	0.1910	0.2800	34	23	50	334
U-0.5_PW-.best.f	15	-27	329	6.8540	0.1820	0.2910	30	23	48	212
U-0.5_def.best.f	16	-48	411	8.5620	0.1930	0.2930	32	24	48	222
U-0.75_A+.best.f	15	-33	557	11.1400	0.2070	0.3060	34	25	50	174
U-0.75_A-.best.f	16	-61	175	3.6460	0.1850	0.2740	31	24	48	209
U-0.75_C+.best.f	17	-41	436	9.0830	0.2090	0.2990	34	28	48	319
U-0.75_C-.best.f	12	-35	358	7.4580	0.1740	0.2910	27	20	48	185

continued on the next page

continued from the previous page

Test label	2	3	4	5	6	7	8	9	10	11
U-0.75_ND+.best.f	17	-47	473	9.8540	0.1990	0.2970	32	24	48	231
U-0.75_ND-.best.f	16	-47	430	8.9580	0.1980	0.2940	32	24	48	212
U-0.75_NW+.best.f	12	-48	377	7.8540	0.1900	0.2930	29	23	48	256
U-0.75_NW-.best.f	13	-47	314	6.5420	0.1910	0.2860	32	25	48	169
U-0.75_PD-.best.f	13	-54	273	5.8090	0.1640	0.2980	27	18	47	195
U-0.75_PW+.best.f	14	-67	181	3.6200	0.1930	0.2830	34	23	50	206
U-0.75_PW-.best.f	15	-26	322	6.7080	0.1850	0.2930	30	24	48	170
U-0.75_def.best.f	16	-47	427	8.8960	0.1980	0.2940	32	24	48	212

Table 2: Utility scores of running Rocchio with different parameters

UB at TREC-11: Batch and Adaptive Filtering

M. Srikanth X. Wu R. Srihari

Department of Computer Science and Engineering
State University of New York at Buffalo,
Amherst, NY 14228-2567

1 Introduction

This is the first time we participated in TREC filtering track. We submitted four runs: two for adaptive filtering, and two for batching filtering. And these runs come from two separate efforts with very different approaches. One effort treats the filtering problems as standard text categorization problems and solves them using Support Vector Machines (SVM). The second effort is a Language Modeling approach to information filtering. Among other things we wanted to use filtering tasks as large scale test cases for two separate frameworks we have been working on for information retrieval. Significant time was spent on putting the components together and limited time on pre-submission performance evaluation.

2 Weighted Margin SVM for Information Filtering

2.1 SVM in text categorization

The standard text categorization problem can be stated as following: given a set of category-labeled documents, the goal is to classify a new document into the predefined categories. Typically each document can belong to multiple categories or no category at all. This is a supervised machine learning problem. And further more, when the categories form a flat structure, each category can be treated as a separated dichotomy problem.

SVM are based on Structural Risk Minimization (SRM) principle from statistical learning theory [7]. In contrast to the Empirical Risk Minimization (ERM) principle which try to find a hypothesis h from a structural complexity fixed hypothesis space, H, that minimizes the training error, the SRM try to find a hypothesis h where the true error of the classifier is minimized. To achieve this, SRM usually tries to shrink the complexity of the hypothesis space H while maintaining a fixed training error. Compare to ERM, SRM is more suited when the training data set is limited.

SVM, the simplest linear form of SRM, is nothing but a maximum margin linear classifier. Given an example (x_i, y_i), and decision hyperplane (w, b), the (functional) margin of example with respect to hyperplane is defined as

$$\gamma_i = y_i(< w \cdot x_i > +b). \tag{1}$$

Note γ_i implies correct classification of (x_i, y_i) if $\gamma_i > 0$. The (functional) margin of training set S (with l examples) with respect to decision hyperplane (w, b) is defined as

$$\gamma = \min_{0 \leq i < l} y_i(< w \cdot x_i > +b), \tag{2}$$

Geometric margin is the functional margin derived by $\|w\|$. The maximum margin hyperplane given training set S is thus defined as the hyperplane with respect to which the training set has maximum geometric margin.

There are two major advantages of SVM. First the learning ability of SVM is independent of the dimensionality of the feature space thus immune from the so-called *curse of dimensionality.* SVM learning process typically focuses on these hard to classify patterns automatically and thus can ignore the extra noise introduced by each additional dimension. Thus by using SVM, the usually computational expensive feature selection steps are not needed. Second, with so-called *kernel tricks*, SVM can be used to learn different discriminant functions by using different kernel functions. So it can learn a linear classifier, polynomial classifier, or radial basis function with just one line change in your source code.

SVM has been used for text categorization [3, 2, 5] and shown to outperform all classical learning methods including neural networks, linear discriminant function and KNN. And this claim is further verified by Y. Yang in her well-cited comparison papers [9, 8]. The reason for its superior performance can be best explained by its compatibility to text categorization problems: the typical document representation method like term frequency and inverted document frequency (TFIDF) weighted vectors results in huge and sparse vector for each document and most text categorization problems are linearly separable (as suggested by the fact that all ohsumed categories and most Reuters categories are linearly separable).

2.2 Filtering as a Text Categorization Problem

While last year batch filtering track can be easily cast into a text categorization problem, it is not possible this year. For one, we don't have enough training examples available for most all the topics, and it is even worse in adaptive filtering track where we have only three positive examples for each topic. Second, the rich information contained in the topic description/narrative that help to shape topic boundary is not readily available for any statistical based machine learning methods. And actually it is our guess that the ability of approaches to make use of the topic description/narrative will differentiate their performance. In the final result, the fact that people can do much better in first 50 categories than in the second 50 categories can be considered as a manifestation of our guess. Since the quality of topic description/narrative is much better in first 50 categories is much better than these of the second 50 categories.

SVM is the winner of the last year batch filtering track. And it is considered as one of major component of our package, to really test out our SVM implementation, we use it to handle the filtering track problem. To do that, there are two problem we have to deal with. First, we have to turn the information stored in the topic description/narratives into some usable information for our SVM learner. Since the only information that SVM learner can make use of is the examples, we have to device a way to generate pseudo examples using topic description/narrative. Second, how our SVM learner make use of these pseudo examples?

2.3 Generating Pseudo Examples

The way we generate pseudo examples from the topic description and narrative is very simple. And there are two different steps. First, we use the description/narrative get the top 30 closest document in TFIDF sense, and label them as probable positive example. Note that they definitely can't be considered as 100% positive examples. Second, we randomly choose 90 documents from the whole training set excluding the existing training examples and top 1000 closest documents to topic description/narrative in TFIDF sense and label them as probable negative examples. This

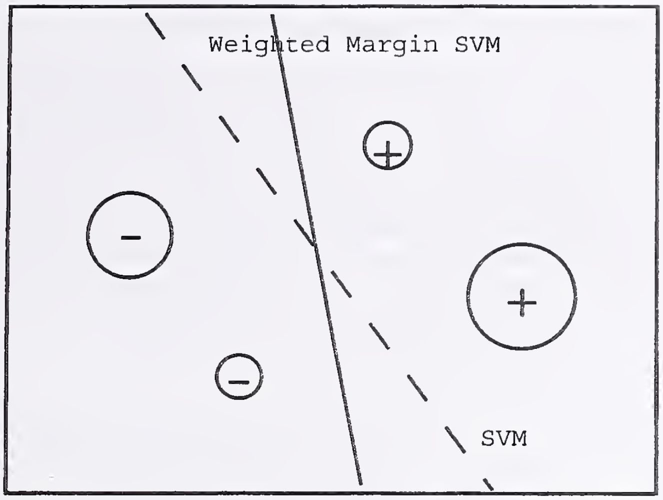


Figure 1: SVM vs Weighted Margin SVM

is extreme simple way to utilize the rich information contained in the topic description/narrative. And the number of probable positive and negative examples we use is set arbitrarily for all the topic without further investigation.

2.4 Weighted Margin SVM

With newly generated probable examples, the question now is how to use them in SVM learning. To be able to use the training set where label is associated with an observation weight, the SVM training and classification algorithms has been modified to handle these additional information available to the learner: and the result of such modification is the Weighted Margin SVM (WMSVM) machine.

The difficulty a classical SVM classifier faces when presented with weighted observation can be best illustrated by the figure 1. We have four pieces of labeled data: two positive and two negative. The size of the circle represents the label reliability. The dashed line represents the hyperplane found by SVM, and the solid line the hyperplane by the Weighted Margin SVM.

In the results submitted for the filtering track, we were using a weaker version of the WMSVM implementation that can handle the weighted soft margin. Since then, we have a stronger WMSVM implementation that can handle both hard margin and soft margin correctly, but we are still working the experiments with this strong version on filtering track. But even with the weaker version WMSVM, along with the simple pseudo example generation, we are able to apply SVM learning on a data set which for some cases, has only a few positive examples.

3 Language Modeling Approach

A model-based approach to information filtering was explored for the second set of submissions to TREC 2002 batch and adaptive filtering tasks. Language models are associated with both documents and queries. The initial query model was generated using the topic description. We viewed the training documents for a given query as relevance feedback documents. A new query model is estimated based on the initial query model and the language models estimated for the feedback documents. This method was used for both batch and adaptive filtering.

3.1 Language Modeling and Information Retrieval

Statistical Language Modeling (SLM) has been used in many Natural Language Processing (NLP) tasks including Speech Recognition, Machine Learning and Information Extraction. Recently, Ponte and Croft [6] proposed language modeling approach to information retrieval. Each document in a document collection is associated with a language model and given a query, documents are ranked based on the probability of their language model generating the query text. Alternatives to query likelihood model have been proposed. Of specific interest here is the method proposed by Lafferty and Zhai [4] where they associate language models to both documents and queries and rank documents based on their model's similarity with the query model. Model similarity was computed using the Kullback-Leibler divergence measure.

The motivation for our language modeling approach to TREC-11 batch and adaptive filtering is Zhai and Lafferty's paper on incorporating relevance feedback in language modeling approaches to information retrieval [10]. Their proposal was in the Query-Document Model similarity approach to information retrieval Given a set of feedback documents $F = \{d_1, d_2, \cdots, d_n\}$ for query Q, they estimate a feedback model $\hat{\theta}_F$ based on the feedback document F and use it to update the query model $\hat{\theta}_Q$ to $\hat{\theta}_{Q'}$ by

$$\hat{\theta}_{Q'} = (1 - \alpha)\,\hat{\theta}_Q + \alpha\,\hat{\theta}_F \tag{3}$$

where α is the interpolation parameter. Two different strategies were proposed for feedback model estimation: a generative model of feedback documents and a model with minimum divergence over feedback documents. We used the later in our language modeling approach to information filtering. Zhang and Callan [11] used a method similar to the generative model for feedback documents in their TREC 2001 adaptive filtering submission. They used language modeling techniques in updating terms and term weights in their query representation.

In the divergence minimization approach, the feedback model is estimated to satisfy two conditions: (1) that it is "closer" to the feedback documents and (2) it is "farther" from the corpus model. The second condition ensures that the effect of language and domain characteristics common to feedback documents do not generalize the new query model and move it off topic. The feedback model is selected to be the one which minimizes

$$D_e(\theta; F, C) = \frac{1}{|F|}\sum_{i=1}^{n} D(\theta||\hat{\theta}_{d_i}) - \lambda\, D(\theta||p(\cdot|C)) \tag{4}$$

where $p(\cdot|C)$ is the corpus probability distribution and λ is the feedback parameter.

3.2 Language Modeling approach to Information Filtering

Given a query Q, two language models are estimated: (1) positive or on-topic language model, $\hat{\theta}_P$, and (2) negative or off-topic language model, $\hat{\theta}_N$. The initial positive and negative models are estimated from the topic description. These models are updated based on the training data available for each query. The positive examples $F_p = \{d_{p_1}, d_{p_2}, \cdots, d_{p_{|F_p|}}\}$ are used to update the positive model $\hat{\theta}_P$ using the feedback model generated by minimizing (5) which is similar to (4).

$$D_n(\theta_{F_p}; F_p, C) = \frac{1}{|F_p|}\sum_{d \in F_p} D(\theta_{F_p}||\hat{\theta}_d) - \lambda\, D(\theta_{F_p}||p(\cdot|C) \tag{5}$$

While the negative language model $\hat{\theta}_N$ can be used instead of the corpus probabilities in (4), we have used the corpus model since it is a better representation of what is not in topic. A negative feedback model is generated by minimizing

$$D_n(\theta_{F_n}; F_n, C) = \frac{1}{|F_n|} \sum_{d \in F_n} D(\theta_{F_n} || \hat{\theta}_d) - \lambda\, D(\theta_{F_n} || \hat{\theta}_P) \tag{6}$$

where $F_n = \{d_{n1}, d_{n2}, \cdots, d_{n|F_n|}\}$ is the set of negative examples. Here one is interested in a negative feedback model that is "closer" to negative examples by "farther" from positive language model. Unlike [10] who used a Dirichlet smoothing in estimating document language model, $\hat{\theta}_d$, we used a mixture model with fixed weights for document and corpus statistics.

$$p(w|\hat{\theta}_d) = \gamma\, p(w|d) + (1 - \gamma)\, p(w|C) \tag{7}$$

where γ was set to 0.6.

The positive and negative topic models are updated by

$$\hat{\theta}_{P'} = (1 - \alpha_1)\, \hat{\theta}_P + \alpha_1\, \theta_{F_p} \tag{8}$$

$$\hat{\theta}_{N'} = (1 - \alpha_2)\, \hat{\theta}_N + \alpha_2\, \theta_{F_n}. \tag{9}$$

Given a test document, its language model is first estimated using (7). Its score is determined by the ratio of its divergence from positive and negative models

$$score(\hat{\theta}_d; \hat{\theta}_P, \hat{\theta}_N) = D(\hat{\theta}_d || \hat{\theta}_P) / D(\hat{\theta}_d || \hat{\theta}_N). \tag{10}$$

Document scores are thresholded to make the binary classification decision. Thresholds were estimated based on score distribution in the training set similar to the method used by [1]. The score of relevant documents are assumed to be normally distributed and the top non-relevant documents are exponentially distributed. The utility score is optimized to obtain a closed form solution for the threshold.

For adaptive filtering, there are no negative examples and hence the initial negative model is the corpus model. The above method is followed to classify documents. When a document is deemed relevant for a query by the system, its relevance judgment is fetched to update the language models. If the document was judged relevant to the topic, the positive model is updated and if it was deemed not relevant the negative model was updated. While the models can be updated irrespective of the relevance of the document, for our TREC submission, we only updated one model at a time. The score threshold is updated before moving on to the next document.

Some inplementation specific details and observation on our system's performance are given here. In our implementation,

- Document and queries were stemmed and stop words were removed.
- Only the topic description was used in generating the initial topic model generation
- While computing the score, terms with probability less than 0.0001 were ignored.
- Instead of top non-relevant document scores, all non-relevant document scores were used in computing the threshold.
- In adaptive filtering, the corpus statistics were not updated as test documents are processed. The training document collection was used to estimate the corpus model.
- In adaptive filtering, documents whose relevance is not known is assumed to be not relevant to the topic and is used in updating the negative topic model.

4 Observations and Conclusion

The results we submitted to filtering track using either methods is not impressive. There are couple reasons for that. With regards to our Weighted SVM submission, besides the fact we didn't use a stronger version WMSVM, the naive way of making use of topic description/narrative probably killed us. This is partially supported by the different performance difference between us and best performance for topic: in the first 50 topic where description/narrative is rich in content, we lag far behind the best performer. But on the second 50 topic, where the topic description/narrative is not as good, we are a little bit closer to the best performer overall.

With regards to the language modeling approach, our initial analysis suggests that the thresholding technique used seems to favor high recall taking our system closer to an "allow-all" classifier. This could have been due to the characteristics of the measure we used for scoring documents. Using all non-relevant documents in our threshold computation seems to have affected the thresholding processes. This was compounded by the assumption of documents with unknown relevance as non-relevant.

Either methods, we believe, have scope for further improvement with respect to their application in information filtering. We expect the stronger version of WMSVM to perform better. In addition we are exploring better ways to generate pseudo examples from topic description/narratives. At same time, a couple parameter, such as the observation weight for each pseudo examples and the number of pseudo examples, can be tuned to improve filtering performance.

References

[1] A. Arampatzis, J. Beney, C Koster, and T van der Weide. Incrementality, half-life, and threshold optimzation for adaptive document filtering. In E. M. Voorhees and D. K. Harman, editors, *TREC 2000*, Gaithersburg, MD, 2000.

[2] R. Cooley. Classification of news stories using support vector machines. In *Proceedings of IJCAI'99 Workshop on Text Mining*, Stockholm, Sweden, 1999.

[3] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[4] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *SIGIR01*, pages 111–119, 2001.

[5] David Lewis. Applying support vector machines to the trec-2001 batch filtering and routing tasks. In *NIST Special Publication 500-250: The Tenth Text REtrieval Conference*, pages 286–292, 2001.

[6] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *SIGIR98*, pages 275–281. ACM, New York, 1998.

[7] Vladimir N. Vapnik. *The nature of statistical learning theory, 2nd Edition.* Springer Verlag, Heidelberg, DE, 1999.

[8] Y. Yang and X. Liu. A re-examination of text categorization methods. In *22nd Annual International SIGIR*, pages 42–49, Berkley, August 1999.

[9] Yiming Yang. An evaluation of statistical approaches to text categorization. *Information Retrieval*, 1(1-2):69–90, 1999.

[10] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *CIKM*, pages 403–410, 2001.

[11] Y. Zhang and J. Callan. The bias problem and language models in adaptive filtering. In E. M. Voorhees and D. K. Harman, editors, *TREC 2001*, pages 78–83, Gaithersburg, MD, 2001.

A Crude Cut at Query Expansion

Phil Rennert
StreamSage, Inc.
phil.rennert@streamsage.com

I had planned to approach the Novelty task by using machine learning methods to fill templates. However, the paucity of training topics (three: I invalidated 379 due to the large inter-assessor disagreement) and the sparseness of relevant-but-not-new sentences would not support this approach.

To do what I could with the training data, I experimented with query expansion. For search terms, I took all the nouns in the title; and added two categories of expansion:

> 1) Multi-word units. StreamSage has a corpus of New York Times news stories, from which we've extracted all multi-word units (words which occur together in sequence) meeting certain grammatical and frequency criteria. I selected all MWUs containing a title noun, and added all other nouns in that MWU as search terms.
>
> 2) Noun phrases in the description (desc, not desc1). Any nouns occurring in the same noun phrase with a title noun were added as search terms.

Then I returned any sentence containing a search term as relevant.

To determine which sentences were new, I used a pure length criterion. The shortest sentences were returned relevant-but-not-new, in the same proportion as in the training data (where 5-10% of the relevant sentences were assessed as not-new).

I hope to do more next year, with 50 training topics available.

Question Answering:
CNLP at the TREC-2002 Question Answering Track

Anne R. Diekema, Jiangping Chen, Nancy McCracken, Necati Ercan Ozgencil, Mary D. Taffet,
Ozgur Yilmazel, and Elizabeth D. Liddy

Center for Natural Language Processing
Syracuse University, School of Information Studies4-206 Center for Science and Technology
Syracuse, NY 1324-4100
www.cnlp.org
{diekemar,jchen06,njm,neozgenc,mdtaffet,oyilmaz,liddy}@syr.edu

Abstract
This paper describes the retrieval experiments for the main task and list task of the TREC-2002 question-answering track. The question answering system described automatically finds answers to questions in a large document collection. The system uses a two-stage retrieval approach to answer finding based on matching of named entities, linguistic patterns, keywords, and the use of a new inference module. In answering a question, the system carries out a detailed query analysis that produces a logical query representation, an indication of the question focus, and answer clue words.

1. Introduction

The Center for Natural Language Processing (CNLP) participated in the main task and the list task of the Question Answering track. The main task required answering 500 short fact-based questions, which have been extracted by NIST from MSNSearch and AskJeeves logs. Unlike previous years, the answer had to be exact, the answer string containing nothing but the answer itself. Also, unlike previous years, the answers to all 500 questions had to be ordered by answer confidence rather than by question number. This means that the answers that the system is most confident about should be ranked first, and the least confident should be ranked last. The scoring (see section 3) reflects a system's ability to determine how accurate a certain answer is. Not all questions had a known answer in the collection. Unanswerable questions have to be identified as such by the system to be counted correct.

The list task required answering 25 short fact-based list questions. List questions include an indication as to how many unique answer instances are needed to answer the question. A response to a list task question consisted of an unordered list of exact answers. The different answer instances could be found within single documents or across multiple documents, or a combination of both. Not all questions have all required answer instances in the collection.

This year there was a new document collection for the Question Answering track. Answers to both main task and list task questions had to be retrieved automatically from 1,033,461 documents from the following three sources: AP newswire, 1998-2000, New York Times newswire, 1998-2000, and Xinhua News Agency, 1996-2000.

2. System Overview

The CNLP question-answering system consists of four different processes: question processing, document processing, paragraph finding, and answer finding. The first three processes are similar to last year's system. [1] Changes were made to the answer finding module to adapt the system to the track's new requirements and to incorporate our new inference module.

2.1 Question processing

Question processing has two major parts - conversion of questions into a logical query representation and question focus recognition. Our L2L (Language-to-Logic) module was used this year to convert the query into a logical representation suitable for keyword matching and weighting in our answer finder module. Question focus recognition is performed in order to identify the type of answer expected, extraction of the number of answers required (used for the list task only), and assignment of a confidence level.

2.2 Document processing and paragraph finding

For document retrieval, we used the ranked document list as provided by NIST. The top 200 documents from the list for each question were extracted from the TREC collection as the source documents for paragraph finding. In the paragraph finding stage, we aim to select the most relevant paragraphs from the top 200 retrieved documents from the first stage retrieval step. Paragraph selection was based on keyword occurrences in the paragraphs. Paragraph detection is based on orthographic clues.

2.3 Answer finding

Four different strategies were applied to find the correct answers to questions. The four strategies were based on entity extraction, inference, answer patterns, and answer context, respectively. The latter three are still under development and did not contribute much to answer finding for TREC-2002. A triage program was developed to classify questions into different answer strategies based on their question type, question focus and the number of keywords.

2.3.1 Entity based approach

The entity-based strategy used the tagged paragraphs from the paragraph finding stage and identified different paragraph windows (different keyword combinations) within each paragraph. A weighting scheme was used to identify the most promising paragraph window for each paragraph. These paragraph windows were then used to find answer candidates based on the question focus. All answer candidates were weighted and the top one was selected. The strategy is similar to previous years with the addition of a new function for assigning an answer confidence score.

For each answer candidate, the system assigned a confidence score to indicate the systems' confidence regarding answer correctness. The confidence score was determined by the following factors: 1) number of keywords in the same sentence, 2) question focus, 3) categorization confidence, and 4) the presence of other answer candidates in the same sentence. A threshold was determined for the confidence judgment score. If a question had a top answer whose confidence score was below the threshold, the question would be marked as having no answer.

2.3.2 Inference based approach

The inference-based approach used the results of our existing event extraction system on text to assist in finding exact answers for queries that involve events, indicated by a verb. These extractions were saved as information frames, and we implemented an inference engine to search for extracted information and to use some simple forms of linguistic inference. While this question approach was designed to answer queries where identifying events was important to the answer, we also could utilize the inference engine to answer questions that needed two or more pieces of information to find the answer.

This approach starts with our existing generic entity and event extraction system. This system extracts event/agent/object information from sentences and also relation extraction about entities, primarily named entities, such as location, point-in-time and characteristic. The generic extraction is implemented using shallow parsing rules. To use generic extraction for Q&A, we processed the queries with the generic extraction system as well by adding shallow parsing rules for query forms and generating an answer template that represents the form of the answer as a generic extraction with the "exact answer" slot filled in with an unknown variable. Informally, in order to answer "When did Hawaii become a state?", we formulate a template "Hawaii became a state in time ?X", where ?X is a variable. For the "What <type of thing>" questions, we would generate a two-part answer template. For example, in "What king signed the Magna Carta?", we would generate both "?X signed the Magna Carta" and "?X is a king".

2.3.2.1 Rule patterns for queries

In order to analyze queries, we wrote shallow parsing rules that could recognize the query patterns. We describe a selection of those patterns here. Note that the query rules did not have to indicate additional qualifying phrases as those would be added by the generic extraction.

Patterns with possibly significant verb phrases:	
when do <nounphrase> <verbphrase> <nounphrase>	When did George Orwell write Animal Farm?
when do <nounphrase> <verbphrase>	When did Mt. St. Helens erupt?
Where do <nounphrase> <verbphrase>	Where did the ukulele originate?
who <verbphrase> <nounphrase>	Who invented baseball?
what do <nounphrase> <verbphrase>	What do bats eat?
Patterns with what (or which) <typeofthing>:	
what <typeofthing> do <nounphrase> <verbphrase>	What flower did Vincent van Gogh paint?
what <typeofthing> be <nounphrase> in	What hemisphere is the Philippines in?
what <typeofthing> be <nounphrase>	What color is a poison arrow frog?
what <typeofthing> be <nounphrase> <prepphrase>	What gasses are in the troposphere?
what <typeofthing> <verbphrase> <nounphrase>	What American composer wrote the music for "West Side Story"?
what <typeofthing> <verbphrase> <prepphrase>	What currency is used in Australia?

Table 1. Rule patterns for queries.

2.3.2.2 Query answer templates

When a query is processed by one of the query rules, one or more templates is generated to use in finding answers in the extraction database. An answer template is a frame in the same format as the frames in the extraction database. For each query, the frames are generated in the format of possible answers to the query, except that the unknown part is given as a variable, represented as the string ?X. (For TREC queries, we only needed to generate answer templates with one unknown variable.) It is the job of the inference engine to fill in a value for the variables, which will be an exact answer to the query.

In general, "when" queries ask for a property that is called "point-in-time" by the extraction system. For "do verb" forms, the first nounphrase in a sentence with an active verb is assumed to be the agent of that event.

when do <nounphrase> <verbphrase> <nounphrase> (When did George Orwell write Animal Farm?)

```
        event = do write
          agent  = George Orwell
          object  = Animal Farm
          point-in-time  = ?X
```

```
when do <nounphrase> <verbphrase>  (When did Mt. St. Helens erupt?)
          event = do erupt
             agent  = Mt. St. Helens
             point-in-time  = ?X
```

In a where query, there is a property "location" for events.

```
where do <nounphrase> <verbphrase>(Where did the ukulele originate?)
          event = do originate
             agent  = ukulele
             location  = ?X
```

Some query patterns are asking for the agent of the object of an event.

```
who <verbphrase> <nounphrase>(Who invented baseball?)
          event = invent
             agent  = ?X
             object  = baseball
```

```
what do <nounphrase> <verbphrase>  (What do bats eat?)
          event = eat
            agent = bats
            object = ?X
```

The "what <typeofthing>" patterns generate two frames for the two pieces of information. The second frame qualifies the answer as to what type of thing it is. The property is called "description" here, and there are several actual extraction properties that can be used to establish that the answer matches this description. Note that this frame is an entity frame.

```
what <typeofthing> do <nounphrase> <verbphrase>  (What flower did Vincent van Gogh paint?)
          event = paint
            agent = Vincent van Gogh
            object = ?X
          entity = ?X
            description = flower
```

The other "What <typeofthing>" query types similarly generate the second frame.

2.3.2.3 Extraction matching with the inference engine

For each query, the answer candidate documents were processed using the generic extraction system. The extractions were put into a database that we call the knowledge base. For answering queries, we then tried to match the template from the query, which is the "goal", with extractions in the knowledge base. We call the matcher the inference engine, but the types of inferencing that we are doing are linguistic in nature. We are not using inference rules that rely on world knowledge.

The inference engine tries to match all the frames representing the goal template. For each frame, it establishes that each attribute of the goal frame is present in the answer frame and that the values of each attribute "match". In order to match values of attributes, the inference engine has several rules to establish a match even if it is not an exact match.

Although we have not shown this in the examples so far, in addition to the string that is kept as the value of an attribute in the extraction frame, some string values also have links to an entity extraction frame. If such a link is present, the inference engine will also check that any additional attributes of that entity are also matched by the answer value. This is used in more complex queries that have additional modifiers.

Although the inference engine tries to match all of the attributes of the goal frame, it uses an abductive inference rule that allows a frame to match even when not all of the attributes are present, but with a lower probability of matching.

Finally, the inference engine has a set of axioms that embody linguistic knowledge about different forms of frames to try to match. If the goal frame has no match in the knowledge base, then these axioms are used to generate new goal frames that are sufficient to establish the answer.

An example of the types of linguistic alternatives is the changing of a goal event frame into an equivalent entity frame where that entity is described by the nominalization of the verb. This rule employs a list of such subject nominalizations.

event = invent	entity = ?X
agent = ?X	description = inventor
object = road traffic cone	modifier = of the road traffic cone

2.3.3 Pattern based approach

The pattern based approach is used for certain types of questions only: acronym, counterpart, definition, famous for, stand for, synonym, why. [1] We developed lexical pattern rules for answer extraction for these special question types. These patterns were used to identify text segments that could possibly provide an answer. Each of the answer identification patterns had its own confidence score indicating the likeliness of that pattern identifying for example, the meaning of a synonym. Unfortunately, the pattern-based approach did not prove effective for the TREC-2002 questions, partly because there were no definition questions this year. However, we find that the pattern based approach is useful in answering student's questions in the aerospace domain in a funded project for NASA.

2.3.4 Context-based approach

The context-based approach deals with those questions for which the system could not determine a question focus, which happens frequently (194 (39%) out of 500). When the system fails to identify a focus, the system attempts to find answers by using the context (the sentence in which the question keywords appear). This approach to answer finding is rather inexact and should be viewed as a last ditch effort.

3. Results

We submitted three runs for the TREC-2002 QA track: one run for the main task and two runs for the list task.

3.1 Main task results

Average over 500 questions	SUT11IR1MT
Confidence-weighted score	0.225
Number wrong	422
Number inexact	5
Number unsupported	9
Number right	64
Precision of recognizing no answer	0.167 (12 / 72)
Recall of recognizing no answer	0.261 (12 / 46)
Questions with rank above the median	47
Questions with rank on the median	427
Questions with rank below the median	26

Table 2. Question answering result for the main task.

The evaluation measure for the main task (see Table 2) is the confidence-weighted score (similar to the uninterpolated average precision measure from information retrieval). The score for an individual question is the number of correct answers up to and including that question divided by the number of questions answered so far. The score for the entire run is the mean of the individual questions' scores. The confidence-weighted score can range from 0 to 1 inclusive, with 1 a perfect score.

3.2 List task results

Average over 25 questions	SUT11IR1LT	SUT11IR1LT2
Average Accuracy	0.11	0.15
Questions with no answer found	17	15
Questions with rank above the median	5	8
Questions with rank on the median	17	15
Questions with rank below the median	3	2

Table 3. Question answering results for the list task.

The evaluation measure for the list task (see Table 3) is accuracy. The score for an individual question is the fraction of unique, correct instances over the target number of instances. The score for the entire run is the mean of the individual questions' scores. Accuracy can range from 0 to 1 inclusive, with 1 a perfect score.

4. Analysis

The analysis centers on the performance of our focus identification module and the contribution of each of the four different answer-finding approaches to question answering.

4.1 Main and list task performance

The large majority of the questions in the main task (84%) and list tasks (68%) were answered incorrectly. The number of questions for which our performance is the same as the median performance (of all participating systems) is close to (427 => 422), or identical (17 => 17, 15 => 15) to the number of questions that we answered incorrectly. These numbers seem to suggest that a lot of systems could not answer most of the questions. Further analysis is needed to determine why this is the case.

4.2 Focus identification

Focus identification is the most important procedure of query processing. It determines what answer strategy will be applied by the system to search for correct answers and also guides answer candidate selection. The question focus analysis is based on main task run (SUT11IR1MT).

The system correctly identified the focus for 301 questions out of 500 (60%), and incorrectly identified the focus for 5 questions (1%). There are 194 questions (39%) for which the system could not determine the focus (see Table 4). In cases where the focus is identified incorrectly no correct answers were found. When we look at the questions for which no focus could be determined at all we see that all the questions were answered incorrectly. These figures show that having a correct focus helps in finding a correct answer but definitely does not guarantee a correct answer.

500 questions	Correct question focus	Incorrect question focus	No determinable question focus
Correct answer	52 (10.4%)	0 (0%)	12 (2.4%)
Incorrect answer	249 (49.8)	5 (1%)	182 (36.4%)
Total	301 (60.2%)	5 (1%)	194 (38.8%)

Table 4. Question focus assignment.

4.3 Answer finding performance

The system applied several answer finding approaches this year, including a new inference module. However, the effort on these new approaches was limited due to the time constraints, leaving them in an earlier stage of development than we would have liked. When we look at the individual contributions of each of the modules (see Table 5) it becomes clear that the system still largely relies on the entity based approach for the identification of correct answers (49 out of 64 = 77%). Only 37 questions were sent to the inference engine, which managed to answer only 4 of them correctly. As pointed out previously, the pattern-based approach did not prove useful for TREC-2002 questions. Only one question was considered answerable by the pattern based approach and this question was answered incorrectly. The context approach, which is the module that handles questions for which there is no focus available, only answered 10 questions correctly out of the 163 that were assigned to this module. However, as a module that handles questions that no other module can handle, it still answered some questions that would otherwise have been lost. There were 8 questions for which no relevant paragraphs were found. These questions were deemed "unanswerable". This proved correct for only one of them.

	Correct answers	Inexact answers	Unsupported answers	Wrong answers	Total
Entity based approach	49	5	9	218	281
Inference approach	4			33	37
Pattern based approach				1	1
Context based approach	10			163	173
No paragraphs found	1			7	8
Total	64	5	9	422	500

Table 5. Answer finding performance.

5. Conclusions and further research

It appears that most of the questions were not answered correctly by our system and that this is a common problem among participating systems. Analysis of the focus assignment module showed that having a correct focus helps in finding a correct answer but does not guarantee a correct answer. This suggests that improving the focus program to capture more question foci should not be the main center of our future research but rather we have to find other strategies to increase the number of questions we can answer correctly. Analysis of the four different answer finding approaches showed that the three modules other than the entity based module did not contribute much to finding correct answers. However, the reason for this could be that they are at an early stage in development. We will concentrate on further development of these modules.

References

[1] Chen, J., Diekema, A.R., Taffet, M.D., McCracken, N., Ozgencil, N. Ercan, Yilmazel, O., Liddy, E.D. (2002) *Question Answering : CNLP at the TREC-10 Question Answering Track*. In: The Tenth Text REtrieval Conference (TREC-2001). National Institute of Standards and Technology, Gaithersburg, MD., pp. 485-494.

Example Based Text Matching Methodology for Routing Tasks

Ari Visa, Jarmo Toivonen, Tomi Vesanen, Jarno Mäkinen
Tampere University of Technology
P.O. Box 553
FIN-33101 Tampere, Finland
{ari.visa, jarmo.toivonen, tomi.vesanen, jarno.makinen}@cs.tut.fi

Barbro Back
Åbo Akademi University
Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland
barbro.back@abo.fi

Hannu Vanharanta
Pori School of Technology and Economics
P.O. Box 300, FIN-28101 Pori, Finland
hannu.vanharanta@pori.tut.fi

Abstract

We present two variations of a prototype based text matching methodology used in the Routing Sub-Task of TREC 2002 Filtering Track. The methodology examines text on the word level. It is based on word coding and examines the distributions of these codes using document histograms.

This research is supported by TEKES, the National Technology Agency of Finland (grant number 40943/99). The support is gratefully acknowledged.

1 Introduction

A common approach to topic detection and tracking is the usage of keywords, especially, in context of Dewey Decimal Classification [2, 1] that is used in United States to classify books. The approach is based on assumption that keywords given by authors or indexers characterize the text well. This may be true, but then one neglects the accuracy. There are also many automatic indexing approaches. A more accurate method is to use all the words of a document and the frequency distribution of words, but the comparison of frequency distributions is a complicated task. Some theories say that the rare words in the word frequency histograms distinguish documents [5]. Traditionally, information retrieval has roughly been based on a fixed list of index terms [5, 3], or vector space models [9, 8]. The latter ones miss the information of co-occurrences of words. There are techniques that are capable of considering the co-occurrences of words, as latent semantic analysis [6] but they are computationally heavy.

Commonly in filtering, documents are preprocessed with tokenizers, stemmers and stopword lists. Using these methods the processing of the documents become more simple for document classification methods. Next step is to construct feature vectors for documents. The value of the feature is usually based on its significance in the document. Traditionally this is done by using term frequencies and inverse document frequencies. Last year results of TREC 2001 filtering track show that using Support Vector Machine (SVM) for classification can give good results for the routing tasks [7, 4].

In this paper, we present our methodology briefly and concentrate on tests of content-based topic classification, which is highly attractive in text mining. The evolution of the methodology has been earlier discussed in several publications [10, 12, 11]. In the second chapter the applied methodology is described. In the third chapter the experiment with the Reuters database and execution times are described. Finally, the methodology and the results are discussed.

2 Methodology

The methodology used in our runs examines now the documents on the word level. The runs were designed so that the basic principles were kept the same. On the detailed level variation in the methods was added in order to test the robustness of the basic ideas.

2.1 Filtering

The original text was first preprocessed, extra spaces and carriage returns were omitted, and single words were separated with single spaces. With the Reuters database, the preprocessing included selecting the allowed XML fields and removal of the XML tags. For the Visa1T11 run a stopword list was created. Words which were common to the most of the topics where chosen into the stopword list. If the word occurred at least in 75 different topic it was chosen to the list. These words were regarded meaningless to the topic identification. For the Visa2T11 run the text was stemmed with the Porter stemmer.

2.2 Word quantization in Visa1T11

The filtered text was translated into a suitable form for encoding purposes. The encoding of words is a wide subject and there are several approaches for doing it. The word can be recognized and replaced with a code. This approach is sensitive to new words. The succeeding words can be replaced with a code. This method is language sensitive. Each word can be analyzed character by character and based on the characters a key entry to a code table is calculated. This approach is sensitive to capital letters and conjugation if the code table is not arranged in a special way.

The last alternative was selected, because it is accurate and suitable for statistical analysis. A word w was transformed into a number in the following manner:

$$y = \sum_{i=0}^{L-1} k^i * c_{L-i} \tag{1}$$

where L is the length of the character string (the word), c_i is the ASCII value of a character within a word w, and k is a constant.

Example: word is "**c a t**".

$$y = k^2 * ASCII(c) + k * ASCII(a) + ASCII(t) \tag{2}$$

The encoding algorithm produces a different number for each different word, only the same word can have an equal number. After each word has been converted to a code number, we consider the distribution of the code numbers of the words.

The representation of word coded numbers was floating point number. Floating point numbers in our system use a radix of two. Mantissa can have values from $[0.5, 1[$. The representation of floating point number:

$$mantissa * 2^{exponent} \tag{3}$$

Our word coding gives only positive numbers so sign is always positive. The mantissa has information of the beginning of the word and the exponent has information about the length of the word. The quantization of the words uses the values of the mantissa and the exponent. The range of the mantissa is divided to N equal size classes. The exponent is divided to size M classes. The mantissa class number and the exponent class number are used in the calculation of the word class number. Possible number of the mantissa classes of the word varies from 1 to N. The actual word class number is calculated in the following manner:

$$\begin{aligned} word\ class\ number &= \lfloor n \rfloor + (N * \lfloor m \rfloor) \\ n &= (y_{mantissa} - 0.5) * N * 2 \\ m &= \frac{y_{exponent}}{M}, \end{aligned} \tag{4}$$

where n is the mantissa class number of the word and m is the exponent class number of the word. N is the quantization accuracy of the mantissa, M is the quantization step of the exponent and y is the word coded to floating point number with formula 1.

Following example shows how the word coding and word class number generation is done to the word "trec". First the word is converted with word coding formula 1 to a number. Number is represented in floating point format where the radix is two. With formula 4 the word number is converted to a word class number, now N is 35000, M is 24, and k is 256.

$$\begin{aligned} y &= k^3 * ASCII(t) + k^2 * ASCII(r) + k^1 * ASCII(e) + k^0 * ASCII(c) \\ &= 0.909741090144962 * 2^{31} \end{aligned}$$

$$\begin{aligned} word\ class\ number &= \lfloor (y_{mantissa} - 0.5) * N * 2 \rfloor + (N * \lfloor y_{exponent}/M \rfloor) \\ &= 28681 + (35000 * 1) \\ &= 63681 \end{aligned} \tag{5}$$

2.3 Word quantization in Visa2T11

In the Visa2T11 run the word coding is a variation of the Visa1T11 word coding. Now, the alphabet of the training data is first determined from filtered and Porter stemmed training documents. The letters are put into order of their frequencies in the training data. The most frequent letter gets letter code 1, the second 2, and so on. If the letter does not appear in the training documents, it is given letter code 0. These letter codes are now used in formula 1 to replace the ASCII values.

Next, all the words of the training data are converted to word codes and their frequencies are counted. The word-frequency list is sorted according to the word code number. The word codes are classified to C classes using a simple classification scheme. The biggest gap between two succeeding word codes is first found and a class boundary is put between them. Then the sum of frequencies of words in the two new classes are counted. The class with most words is divided into two classes where the gap between two succeeding word codes is the biggest. This method is repeated until there are C classes. The class boundary information and the word codes are used in creating class numbers for the words of the documents.

2.4 Test document to histogram

When examining a single test document, we create a histogram of the word code numbers of the document. The filtered text from a test document is encoded word by word. Each word number is quantized using the word quantization method of the run. The quantization value is determined, an accumulator corresponding to the value is increased, and thus a word histogram A_w is created. The histogram A_w is finally normalized by the length of the histogram vector. The process of converting a document to a histogram is illustrated in Fig. 1. The histogram contains information about the words of the document in a numerical form. This histogram is used in the TREC Routing process to find the best topic for each test document. The diffence or distance between a single test document histogram and the histogram representing the topic can be calculated using different metrics. Among the most simple and effective metrics there are the Euclidean distance and the cosine distance.

With the histograms derived from all the documents in the test database we can compare and analyze the text of the single documents on the word level against the relevant texts of each topic. Note, that it is not necessary to have any prior knowledge of the actual text documents to use these methods. No linguistic methods, other than the Porter stemming, are used in the process.

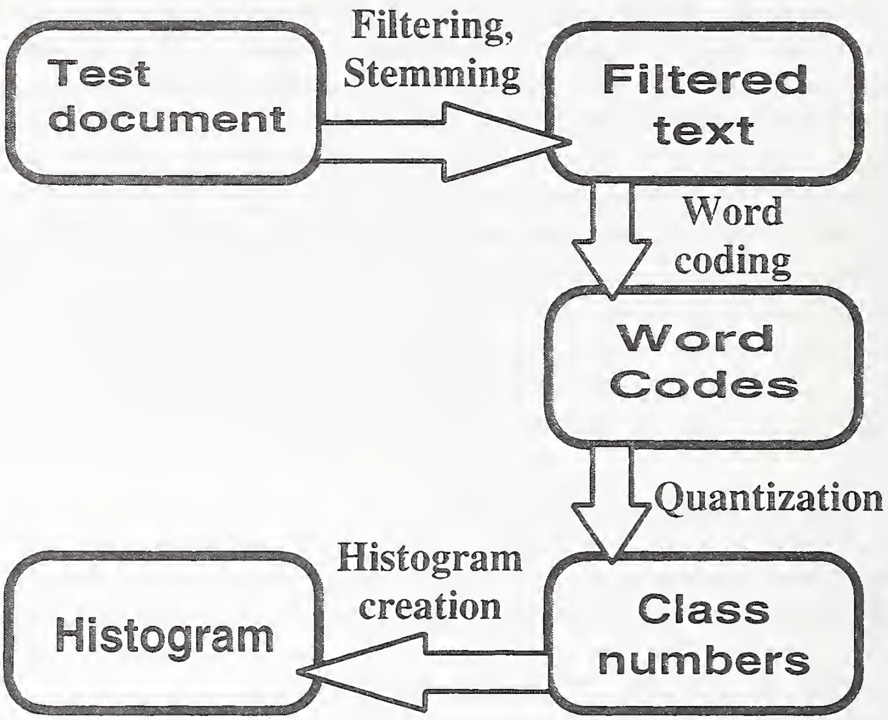


Figure 1: Process of converting document to histogram.

3 Runs with Reuters database

All the relevant documents to a certain topic from the training data were concatenated to one topic document. This document consists of all given relevant text documents classified to that topic. This document was used to define the topic. The information of irrelevant documents to the topic were not used in runs. Every topic document was converted to a normalized topic histogram. All test documents were also transformed to individual histograms and normalized to vector length one. In the two runs we used the methods described in sections 2.2 and 2.3.

Every test document histogram was compared with every topic histogram. The distance metric used in run Visa1T11 was the Euclidean distance. In run Visa2T11 the used distance metric was cosine distance. A topic best-match file was created for each topic. Each test document's four best matching (smallest distance) topics were determined. For these four topics, the ID number of the test document and its distance to the topic were put in the best-match file. From these files the top 1000 documents with the closest distance to the topic were selected for the result file. Our methods gave results which where close to the average level of all participating methods. Visa2T11 gave slightly better results than Visa1T11.

3.1 Execution times

The applied methodology is very fast even with a database as large as the Reuters database. In table 1 we present the execution times we calculated for the two runs. Making histograms execution time consists of creating the word histograms for the test documents. The comparing execution times are the times that it took to compare the test histograms with the topic histograms and to find the four closest topics for each test histogram.

Table 1: Execution times rounded up to the nearest hour.

	Making histograms	Comparing	Altogether
Visa1T11	1 h	6 h	7 h
Visa2T11	6 h	7 h	13 h

The computer used in the experiments was a PC with a Intel® 550 MHz Pentium® III processor and 128 Mb of memory. The operating system was Slackware Linux 7.0.0.

4 Discussion on results

There were some general difficulties when using the methodology on the Reuters database. The selection of documents for the given training set turned out to be disadvantageous. Firstly, it seemed that the set was too unevenly distributed in topics for our methodology. When some topics have under ten relevant documents and some hundreds, statistical methods are in trouble. There is not enough information in just few short relevant documents for this type of methods to be successful. Uneven division in topics also lead to give more weight to topics that have more relevant documents.

Secondly, because the training set was from a period of two months, the vocabulary in the relevant documents does not vary enough. The type of methodology we used requires a good set of representative word samples from the whole database. The training set vocabulary was restricted in the sense of yearly cycle, to two months in autumn of 1996. This type of difficulties are, on the other hand, very common in real life tasks.

Also, we had difficulties with the topics 151-200. Our methodology was not doing well in finding relevant test documents for these topics. This was perhaps partly due to our decision of emphasizing accuracy more than generalization. It may also be due to the nature of the artificial topic construction process.

Our runs were designed so that only a basic form of the methodology was used. The methods used are very fast and it seems that we are improving with the accuracy of the methodology. Visa2T11 had 10000 different classes for the words whereas Visa1T11 had about 6000. There was no training for the classification of words in Visa1T11. Because of smaller number of word classes Visa1T11 had one hour faster comparing time. The drawback was slightly poorer results. One interesting issue in advancing even more is how to use the information of the non-relevant documents for a topic to improve the process. Non-relevant documents seemed to be special cases of relevant documents topics. Our methodology can notll use the information of non-relevant document, because only few words can make distinction between relevant and non-relevant document. In future, this could maybe be achieved by giving negative weight to those kind of words.

References

[1] M. Dewey. *A Classification and subject index for cataloguing and arranging the books and pamphlets of a library*. Case, Lockwood & Brainard Co., Amherst, MA, USA, 1876.

[2] M. Dewey. Catalogs and Cataloguing: A Decimal Classification and Subject Index. In *U.S. Bureau of Education Special Report on Public Libraries Part I*, pages 623–648. U.S.G.P.O., Washington DC, USA, 1876.

[3] T. Lahtinen. *Automatic indexing: an approach using an index term corpus and combining linguistic and statistical methods*. PhD thesis, Department of General Linguistics, University of Helsinki, Finland, 2000.

[4] D. D. Lewis. Applying Support Vector Machines to the TREC-2001 Batch and Routing Tasks. In E. Voorhees and D. Harman, editors, *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, NIST Special Publication 500-250, pages 286–292, Gaithersburg, Maryland, USA, November 13–16 2001. Department of Commerce, National Institute of Standards and Technology (NIST).

[5] C. D. Manning and H. Sch¨utze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[6] D. W. Oard and G. Marchionini. A conceptual framework for text filtering. Technical Report CS-TR3643, University of Maryland, May 1996.

[7] S. Robertson and I. Soboroff. The TREC 2001 Filtering Track Report. In E. Voorhees and D. Harman, editors, *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, NIST Special Publication 500-250, pages 26–37, Gaithersburg, Maryland, USA, November 13–16 2001. Department of Commerce, National Institute of Standards and Technology (NIST).

[8] G. Salton. *Automatic Text Processing*. Addison-Wesley, 1989.

[9] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[10] J. Toivonen, A. Visa, T. Vesanen, B. Back, and H. Vanharanta. Validation of Text Clustering Based on Document Contents. In P. Perner, editor, *Proceedings of MLDM 2001, the Second International Workshop on Machine Learning and Data Mining in Pattern Recognition*, number 2123 in Lecture Notes in Artificial Intelligence, pages 184–195, Leipzig, Germany, July 25–27 2001. Springer–Verlag.

[11] A. Visa, J. Toivonen, H. Vanharanta, and B. Back. Contents Matching Defined by Prototypes – Methodology Verification with Books of the Bible. *Journal of Management Information Systems*, 18(4):87–100, 2002.

[12] A. Visa, J. Toivonen, T. Vesanen, and J. M¨akinen. Tampere University of Technology at TREC 2001. In E. Voorhees and D. Harman, editors, *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)*, NIST Special Publication 500-250, pages 495–501, Gaithersburg, Maryland, USA, November 13–16 2001. Department of Commerce, National Institute of Standards and Technology (NIST).

Incremental Learning for Profile Training in Adaptive Document Filtering[*]

Liang Ma, Qunxiu Chen, Shaoping Ma, Min Zhang, Lianhong Cai
State Key Lab of Intelligent Tech. & Sys., CST Dept, Tsinghua University, Beijing 100084, China
Maliang00@mails.tsinghua.edu.cn

Abstract

In this paper, we describe our ideas and related experiments in TREC-11 Adaptive Filtering Track. In the track we focused much on a robust way for effective profile training. We developed an incremental learning method which selects pseudo positive documents in less bias from a few initial positive training documents. We also did some experiments with newly emerged information retrieval model, language model-based retrieval mechanism, to evaluate its performance when used in adaptive filtering task. Related experiment results show the incremental learning method can be helpful for profile training, while the new language model perform not well.

1. Introduction

In adaptive filtering, firstly we do profile training to get an initial profile, then based on this profile we do adaptive profile updating. Most of the research work now focus on the algorithms for adaptive profile updating because of its immediate effect to the performance. While even a perfect adaptive profile updating mechanism will suffer a poor result if starting updating from a biased initial profile. In fact, it is high potential to get a bias initial profile because of insufficient topic features provided by such few initial positive training documents.

A common method for profile training is like this. First use the query constructed from initial positive documents to score the training set, then get all the pseudo positive documents(used to expand initial profile vector) or setup initial profile threshold. In [1], initial profile threshold is set in a rank position of these scored documents. System in [3] select n documents (n=k*m, m is the number of initial positive documents) with the highest score value in the training set to be pseudo positive documents. Then these documents, together with initial positive documents, are used as positive documents to set initial profile vector and threshold.

It is a simple method together with some problems. By this way of one-step learning, the pseudo positive document and profile threshold are totally depend on these highly limited initial positive documents. Thus any bias in training from these initial training documents will lead to amplified bias in pseudo documents and initial profile. Also the fixed number of pseudo documents, usually set by experience in [3], is hard to be determined for various topics.

In TREC-11, we do further research work in profile training to find a better way for un-bias profile training. In the next section, the detail of profile training will be introduced. After that, experiment data and evaluation result, including our experiments on language model, will be listed. At the end of this paper there is a summary.

2. Incremental Learning in Profile Training

2.1 Feature selection for initial profile

We introduce a two–phase selection mechanism for feature selection from positive documents(in this

[*] Supported by the Chinese National Key Foundation Research & Development Plan (Grant G1998030509). Natural Science Foundation No.60223004. and National 863 High Technology Project No. 2001AA114082.

section, term 'document' are called as 'doc'). First we select key features by a step-extend morphological analysis, then get more extended features by incremental learning.

(1) Get key features for initial profile

We extract initial key terms from topic statement and 3 initial positive training documents.

For topic statement, we use a parser[6] and get terms(called TK term) step by step, something unlike the common way which simply extract all the terms once. The idea is explained as follow:

1. From title field, get all the words as key terms and add them to KeyTerm set.
2. From desc field, we only find the words which limit key terms in KeyTerm set and add them to KeyTerm set.
3. From narr field, we do the same process as step 2.

For 3 positive training docs, we statistic the terms in title and text field(after stopword removing) . The terms(called TK term) whose weight are higher than the double of the average term weight and occur in more than 1 doc are selected as key terms.

We combine the TK terms and DK terms to construct a basic profile. Here the DK terms use their statistic weight. We set the average weight of DK terms as the weight of TK terms. For terms from desc and title field, increase their weight with different weight plus(>1).

(2) Incremental learning for initial profile

We use an incremental learning mechanism for more extend features from pseudo positive docs. Different from the common ways only score training set once and select all the pseudo docs(easily cause the bias problem), in our mechanism training set be repeated scored more than once. After each scoring only a small number of pseudo documents who is highly relative to the exist positive docs are selected, and these docs are used to do limited feedback for new profile vector terms. By this step learning process we can decrease the potential bias pseudo positive docs. The detail of learning process is:

1. Define a set U for positive docs(including pseudo positive docs from learning). Here P_u is number of elements in U. Also we get initial profile by process described above.
2. Use current profile to score all the docs in training set, then sort them by their score in descend order. Set AVG_u as average score of all the docs in U and S_{min} as minimal score of docs in U.
3. Select new pseudo positive documents from scored docs and add them to U. Two rules for selecting them:

   Rule 1: If the first P_u docs are all in U, select the No.(P_u +1) doc.

   Rule 2: Else, select lower value among AVG_u and S_{min} as threshold. The docs whose score are higher than threshold are new pseudo docs.
4. Do feedback(for example, Rocchio method) to profile with new selected pseudo positive docs.
5. goto step 2 for next learning. Exit loop if :
   1. if no new pseudo positive docs can be got.
   2. if P_u >n or already kept learning for r times.

2.2 Setup Initial Threshold

With the key features and extend features we create the term vector of initial profile. The initial profile threshold should be set to value that can result in the highest value of T11F. In calculating the T11F, we count all the docs in U as positive docs.

3. Language Model in Adaptive Filtering

Besides the research work in profile training, we have the interest in how well the language model IR model perform in adaptive filtering. The Lemur [7] tool kits is chosen here to support our experiment. As a newly released IR tool kits, it provide a language model-based IR mechanism for relevance score and related feedback methods. We submitted one run simply with the default parameters based on this system. Thinking that other TREC team which also use it in adaptive filtering will deliver a detail report about it, we just run our experiment with the default parameters and make no deep analysis to it.

4. Experiment

4.1 Performance of Incremental Learning

We use Reuter training corpus as training set to do incremental learning for 100 TREC topics, and let the positive training documents for batch filtering as relative documents for our test. The average T11F and T11U score of 100 topics are calculated for performance estimation. We do two training, one by fixed learning used in [3] with different fixed number (see figure 1) and one by incremental learning(see table 1)

In figure 1, the performance of fixed learning decrease as the number of pseudo positive document increase. Though there is a higher score for small fixed pseudo documents, it is no practical use for training because of low recall.

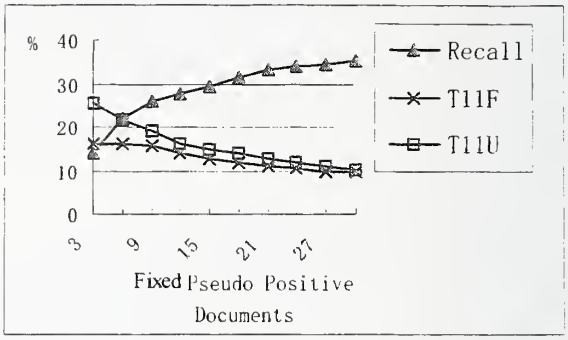


Figure 1. Fixed learning for pseudo positive documents

Table 1: Average Score of All Topics For two methods

Method	Parameters	Pseudo Documents Number	T11F	T11U
Incremental Learning	n=15; r=3	11	21.55%	27.16%
	n=20; r=4	13	20.63%	25.78%
Fixed Learning	n=15	15	12.85%	14.79%
	n=12	12	14.15%	16.31%
	n=9	9	15.74%	19.04%

Two evaluation results of incremental learning run with two set of typical parameters are listed in table 1. With both sets system reach to the similar performance(for example, pseudo documents number). In comparison with the old way, we also list another 3 results by fixed learning which has the approximate pseudo documents number with that in incremental learning. It is obviously that the new incremental learning get the high score both in T11U and T11F.

4.2 Runs Submitted and Evaluation Result

This year we submit 3 runs for adaptive filtering(algorithm for adaptive profile updating is ignored here). To compare the performance of new IR model , traditional Vector Space Model are also used for guideline, and all the runs are optimized by same criteria (T11F). Table 2 show the technology used in each runs.

Table 2: Technology Used in Each Run

Runs	IR model	Score Method	Feedback	Other Process
ThuT11af1	Vector Space Model	TF/IDF(bm25)	Improved Rocchio	Query Expansion

ThuT11af2	Vector Space Model	TF/IDF(bm25)	Improved Rocchio	
ThuT11af3	Language Model	SimpleKL	Mixture Feedback	Query Expansion

Table 3 list the evaluation result for each runs. For the 4 evaluation criteria, we calculate the average value for the first 50 and second 50 topics. Also the average of median value in all the TREC-11 runs is listed for comparison.

Table 3: Average Result of Evaluation for Each Runs

Runs	R101-R150				R151-R200			
	T11U	T11F	Set Precision	Set Recall	T11U	T11F	Set Precision	Set Recall
ThuT11af1	0.395	0.417	0.512	0.367	0.059	0.040	0.038	0.101
ThuT11af2	0.389	0.422	0.474	0.417	0.061	0.052	0.057	0.065
ThuT11af3	0.277	0.337	0.357	0.504	0.052	0.030	0.037	0.060
Avg median	0.381	0.306	0.395	0.286	0.257	0.02	0.031	0.021

From the date above we find the Language Model perform not as well as we expect. Compared to the traditional model, it does not show the predominance in adaptive filtering. We also notice that second 50 topics get better position in all Trec-11 runs than the first 50 topics do, indicating our incremental learning in profile training is more effective for second 50 topics.

5. Summary and Future Work

In TREC-2001 adaptive filtering track, we developed a general method for effective learning in profile training, and did some performance evaluation for language model. Though the language model applied to adaptive filtering wok not well as wish, the new incremental learning method demonstrate its advantage than the old ways. Knowing little on the effect that feedback method we used in incremental learning, we are going to do detailed analysis for different feedback methods in incremental learning.

Reference

[1] C. Zhai, P. Jansen, N. Roma, E. Stoica, D.A. Evans. Optimization in CLARIT TREC-8 Adaptive Filtering Trec 8. In Proceeding of eighth Text Retrieval Conference(TREC-8). NIST

[2] S. Robertson, D.A. Hull. The TREC-9 Filtering Track Final Report. In Proceeding of ninth Text Retrieval Conference(TREC-9). NIST

[3] L. Wu, X. Huang, J. Niu, Y. Guo, Y. Xia . FDU at TREC-10: Filtering, QA, Web and Video Tasks. In Proceeding of tenth Text Retrieval Conference(TREC-10), NIST

[4] A. Arampatzis Unbiased S-D Threshold Optimization, Initial Query Degradation. Decay. and Incrementality. for Adaptive Document Filtering. In Proceeding of tenth Text Retrieval Conference(TREC-10). NIST

[5] S. Robertson. I. Soboroff. The TREC 2001 Filtering Track Report. In Proceeding of tenth Text Retrieval Conference(TREC-10). NIST

[6] Dekang Lin. MiniParser. http://www.cs.ualberta.ca/~lindek/minipar.htm

[7] The Lemur Toolkit for Language Modeling and Information Retrieval. http://www-2.cs.cmu.edu/~lemur/

Expansion-Based Technologies in Finding Relevant and New Information: THU TREC2002 Novelty Track Experiments*

Min Zhang, Ruihua Song, Chuan Lin, Shaoping Ma, Zhe Jiang, Yijiang Jin, Yiqun Liu, Le Zhao
State Key Lab of Intelligent Tech. & Sys., CST Dept, Tsinghua University, Beijing 100084, China
zhangmin99@mails.tsinghua.edu.cn

1 Introduction

This is the first time that Tsinghua University took part in TREC. In this year's novelty track, our basic idea is to find the key factor that help people find relevant and new information on a set of documents with noise. We paid attention to three points: 1. how to get full information from a short sentence; 2. how to complement hidden well-known knowledge to the sentences; 3. how to make the determination of duplication.

Accordingly, expansion-based technologies are the key points. Studies of expansion technologies have been performed on three levels: efficient query expansion based on thesaurus and statistics, replacement-based document expansion, and term-expansion-related duplication elimination strategy based on overlapping measurement.

Besides, two issues have been studied: finding key information in topics, and dynamic result selection. A new IR system has been developed for the task. In the system, four weighting strategies have been implemented: ltn.lnu[1], BM2500[2], FUB1[3], FUB2[3]. It provides both similarity and overlapping measurements, based on term expansion. Comparisons can be made on sentence-to-sentence or sentence-to-pool level.

2 Query Expansion

In the task, it is most possible that relevant sentence is mismatched to the query if we only use the original topic words. Therefore proper query expansion (*QE*) technology is necessary and helpful. Besides thesaurus based *QE* described in section 1 and 2, we proposed a new statistical expansion approach called *local co-occurrence based query expansion*, shown in section 3.

2.1 Using WordNet

Firstly Wordnet[4] is used as the thesaurus to expand query words. Totally three kinds of information were observed in our experiments: hyponyms (descendants), synonyms and coordinated words.

Figure 2.1 shows the effects of *QE* using WordNet hyponyms. Effects of using WordNet synonyms and coordinated words are shown in Table 2.1. In the figure, *hpyo* means to expand all hyponyms and sub-hyponyms of each topic word. And *hypo_1*, *hypo_2* and *hypo_3* refer to expanding words in the direct one or two or three levels of hyponyms respectively. *Hypo_leaf* is to expand hyponyms in leaf nodes of WordNet. Baseline result used long query.

Results show that the more words expanded, the worse the retrieval performance is. All kinds of hyponyms expansion did not help retrieval. Expanding first level hyponyms (average P*R=0.066) makes trivial improvement to the baseline (average P*R = 0.064). Shown in the table, expansion based on synonyms achieves a little improvement in terms of average P*R while it does not help in terms of F-measure.

* Supported by the Chinese National Key Foundation Research & Development Plan (Grant G1998030509), Natural Science Foundation No.60223004, and National 863 High Technology Project No. 2001AA114082.

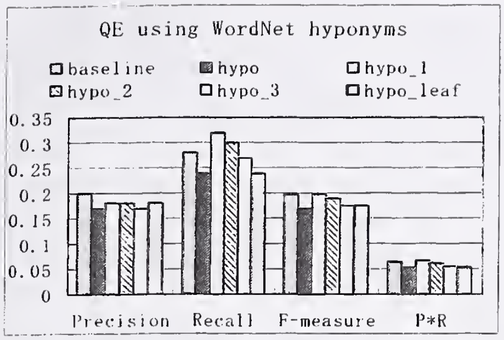


Figure 2.1 Effects of QE with WordNet hyponyms.

Table 2.1 Effects of QE using WordNet synonyms and coordinate words

	P	R	F	P*R
Baseline	0.2	0.28	0.197	0.064
Hypo_1	0.18	0.32	0.197	0.066
Synset	0.17	0.32	0.195	0.068
Coordinate	0.18	0.29	0.189	0.061

P: Average precision R: Average Recall

F: F-measure

P*R: Average Precision*Recall

2.2 Using Dr. Lin Dekang's synonyms dictionary

We also observed the performance by Dr. Lin Dekang's synonyms dictionary[5]. It provides two kinds of synonym dictionaries, based on dependency and mutual information respectively. This *QE* approach works better than the baseline in training set, while makes trivial improvement in test data (see Table 2.2).

Table 2.2 Effects of QE by Dr. Lin Dekang's synonyms dictionary

Ave. Precision	Ave. Recall	F-measure	Ave. P*R
0.18	0.31	0.196	0.067

2.3 QE based on local co-occurrence

We proposed a new statistical expansion approach, which expands terms highly co-occurred in a fixed window size with any of headwords in the relevant document set, called *local co-occurrence expansion (LCE)*. The results are extremely good. Other than most expansion techniques, *LCE* made consistent great progress in terms of both recall and precision. Experimental results are shown in Table 2.3. By using *LCE*, we got 15% and 28% improvement in terms of F-measure and average P*R respectively.

Figure 2.2 gives the overview of query expansion technologies used in our novelty experiments.

Table 2.3 Effects of *QE* by local co-occurrence expansion

	Baseline	LCE
Ave. Precision	0.20	0.21
Ave. Recall	0.28	0.34
F-measure	0.197	0.227
Ave. P*R	0.064	0.081

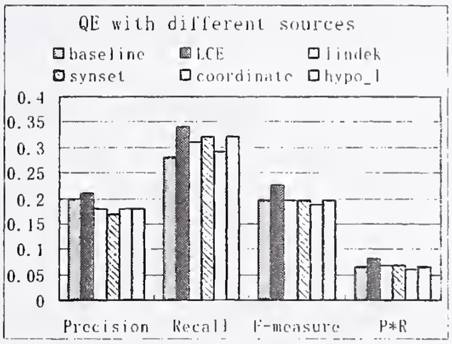


Figure 2.2 Overview of QE experiments

3 Document Expansion

Sometimes, the query mentions a general topic while some relevant documents describe detailed information. For example, the concept of "vehicle" in query is expressed by specific words such as "car", "truck" and "aircraft" in documents. In this case, (1) *QE* may take too many useless words because of aimless of expansion; (2) Setting weights for original and expanded terms is one of the main difficulties in *QE*. Therefore we proposed term expansion in documents (referred as *DE*) to solve the problem.

Other than *QE*, the concept network in WordNet is definitely helpful. We used three levels of hypernyms (ancestor) and their synonyms, referred as *hype_3* in our experiments. The algorithm of document expansion (*DE*) is as following. For each noun in a relevant document, if its 3-level hypernyms include any keyword in query, then replace the noun with the keyword. By doing this, the documents evolve into expanded documents while the query takes no change. Experimental results in Table 3.1 show that *DE* got higher performance than *QE* under the same circumstances. The key point of *DE* is replacement. The keyword and its hyponyms were represented by an identical word, while the keyword and its hyponym were treated as different words in *QE*. Essentially *DE* used the concept space instead of the term space.

Table 3.1 Comparisons between *QE* and *DE*

Method	Ave. Precision	Ave. Recall	F-measure	Ave. P*R
QE (*hypo_3*)	0.14	0.25	0.179	0.057
DE (*hype_3*)	0.18	0.40	0.248	0.079

4 Combination of QE and DE

4.1 Topic Classification by *QE* and *DE*

QE and *DE* are oriented from two aspects of retrieval problem and may work well for different topics. Therefore we classified the topics into two classes according to topic or document characteristics to perform QE or DE respectively, which lead to better performance than either approach.

One intuitive method of classification is topic-oriented. Define fields' similarities in topic: FS_{td} (<title> and <desc>), FS_{tn} (<title> and <narr>) and FS_{dn}(<desc> and <narr>). In our experiments we use the following rules: if $FS_{dn} < \theta_1$ and $(FS_{td}+FS_{dn}-2FS_{tn}) < \theta_2$, then the topic should use *DE* on the topic, otherwise *QE* is performed. The thresholds θ_1 and θ_2 are set according to 0.07 and 0.035.

The other one is document-oriented. Compute the value of: *(# words expanded)/(# words in docs)* for each topic. Only when the value is greater than θ, use *DE*. In our experiments, $\theta = 0.058$.

All the parameters were set according to TREC2002 training examples. It got better performance although the thresholds are not fit for testing data completely. The effects of two approaches are shown in Table 4.1, where *TOTC* and *DOTC* means topic similarity and *DE* oriented topic classification, respectively.

Table 4.1 Effects of topic classification

Method	Ave. Precision	Ave. Recall	Ave. P*R
QE (*LCE*)	0.21	0.34	0.081
DE	0.22	0.28	0.066
TOTC	0.23	0.34	0.087
DOTC	0.23	0.374	0.086

4.2 Result Combination

We've tried several different combination strategies. Here are two that work pretty well. One is called re-ranking (Eq4.1), and another one is called combining inversed rank (Eq4.2). We used Eq2.7 in the experiments. The combined approaches are *QE*(*LCE*) and *DE*. $\lambda \leq 0.3$.

If $Doc_i \in$ result list1 & $Doc_i \in$ list2, then $Sim_i' = \lambda S_{1i}$, $(\lambda > 1)$ else $S' = S_{1i}$	4.1
if $Doc_i \in$ result list1 or $Doc_i \in$ list2, $Sim_i' = \lambda * 1/Rank_{1i} + (1-\lambda) * 1/Rank_{2i}$, $(\lambda < 1)$	4.2

5 Overlap Measurement Strategy Based on Term Expansion

On eliminating repetitive information, rather than concept of similarity, we used the concept of sentence overlapping. It represents the extent of the information taken by one sentence overlapped by another one.

This overlapping measure is unsymmetrical to the compared two sentences. Our experimental results show it is better than the symmetrical measure of similarity. Eq5.1 shows the overlapping of document B by document A, where A is the document preceding B.

$$Overlap B_A = \frac{A \cap B}{B} \qquad 5.1$$

Then the overlapping factor of B is *max{Overlap B$_i$*| document *i* preceding *B*}.

In repetitive information elimination, term expansion was performed. Suppose the two sentences that should be compared are D_1 and D_2, the expanded parts of the original sentences are E_1 and E_2 respectively. Then the basic idea of elimination with term expansion (*TE*) is shown as Eq5.2.

$$OverlapTE(D_1, D_2) = Overlap(D_1, D_2) + \Delta\, Overlap(E_1, D_2) + \Delta\, Overlap(E_2, D_1) \qquad 5.2$$

Table 5.1 shows the result of eliminating repetitive information by using standard qrels of relevant information as the input of the second step. It seems that the dataset used in TREC2002 is not redundant enough for testing the system ability of finding new information.

Table 5.1 Effects of repetition elimination by using qrels of relevant

	Ave precision	Ave recall	Ave P*R
Qrels of relevant info, no elimination	0.91	0.99	0.905
Elimination without TE	0.92	0.99	0.904
Elimination with TE	0.92	0.98	0.900

6 Special Issues

6.1 Finding Keyword in Topics

In Novelty track, all the four domains of the topic can be used to retrieval, while the most useful information is taken by only several keywords. Therefore, finding key information from the topic is an important issue. We classified words in the topic into three classes by statistical learning and rule-based learning: *useful keywords* that contain the most useful words and were used to perform retrieval, *general describing words* that contain little information and were discarded directly and *negative words* that were applied to refine retrieval results.

To remove the topic-free words that contain no more information on describing the topic, two statistical learning methods were performed. Suppose the impact factor of the term is IF_t, terms with impact factor lower than a threshold were general description words. IF_i can be calculated by the two approaches:

$IF_i = qtf_i \,/\, sum_i$ 6.1	$IF_i = tf_i \,/\, n_i$ 6.2

Where qtf_i is the term frequency for t_i in the topic, sum_i is the summation of qtf_i in past TREC queries, tf_i is the term frequency in relevant documents and n_i is the number of documents that the term occurs.

6.2 Dynamic Result Selection

In general information retrieval experiments, the system returns fixed number of results to all the topics. In most cases, however, different topic has different number of relevant documents. Therefore, *how many is enough* is an important issue. We give the algorithm to select the documents whose similarity and rank fit in with the thresholds. Figure 6.1 and 6.2 show the effects of dynamic result selection.

7 Runs Submitted

Table 7.1 show the runs we submitted in novelty experiments, where *DOTC, TOTC* and *QE*(*LCE*) have the same definition of Table 4.1. *Comb_QE_DE* is the combining inversed rank of *QE* and *DE*. The first step results of above four results are got by Okapi system. And the last result is got by our new system with short query. All the second step results were got by the new system.

Figure 6.1 Result number deduction

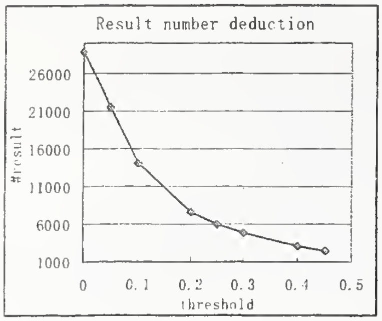
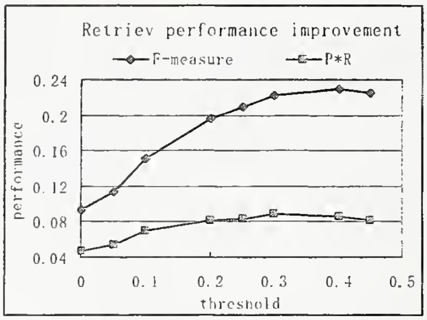


Figure 6.2 Retrieval performance improvement



	Finding relevant information			Elimination repetitive information		
	Ave P	Ave R	Ave P*R	Ave P	Ave R	Ave P*R
Thunv1. DOTC	0.23	0.34	0.086	0.22	0.30	0.073
Thunv2. TOTC	0.23	0.34	0.087	0.23	0.29	0.074
Thunv3. Comb_QE_DE	0.20	0.41	0.088	0.20	0.35	0.073
Thunv4. QE(LCE)	0.21	0.34	0.081	0.21	0.28	0.067
Thunv5. New System	0.19	0.35	0.066	0.18	0.31	0.060

Table 7.1 Submitted runs and evaluation results of Tsinghua University in TREC2002 novelty Track

8 Conclusion and Discussion

In this year's TREC experiments, we mainly focused on the expansion-related technologies. Besides thesaurus based QE, which made only a little progress, we studied a new statistical expansion approach, called *local co-occurrence expansion.* The results are extremely good. It made consistent great progress not only in recall but also in precision. Furthermore, we proposed a novel document term expansion (*DE*) approach. Experimental results proofed encouraging effect of DE. Combinations of *QE* and *DE* by topic classification lead to better performance than either approach. On eliminating repetitive information, rather than concept of similarity, we used the concept of overlap with term expansion. Unfortunately however, it did not take improvement in the experiments.

However, it seems that the dataset used in TREC2002 is not redundant enough for testing the system ability of finding new information, which may influence the conclusion of effectiveness of different approaches. We still take an optimistic view of redundancy elimination technology based on term expansion and overlap measurement.

References

[1] Ricardo Baeza-Yates and Berthier Ribeiro-Neto, Modern Information Retrieval. Addison Wesley. 1999

[2] Robertson. S. E.. and Walker, S. (1999). Okapi/Keenbow at TREC-8. In *TREC-8.*

[3] Gianni Amati. Claudio Carpineto and Giovanni Romano. FUB at TREC-10 Web Track: A probabilistic framework for topic relevance term weighting. In TREC-10.

[4] George Miller. WordNet: An on-line lexical database. International Journal of Lexicography. 3(4). 1990

[5] Dekang Lin. MiniParser. http://www.cs.ualberta.ca/~lindek/minipar.htm.

THU TREC2002 Web Track Experiments*

Min Zhang, Ruihua Song, Chuan Lin, Shaoping Ma, Zhe Jiang, Yijiang Jin, Yiqun Liu, Le Zhao
State Key Lab of Intelligent Tech. & Sys., CST Dept, Tsinghua University, Beijing 100084, China
zhangmin99@mails.tsinghua.edu.cn

1 Introduction

Anchor text has been proofed efficient in former TREC experiments on homepage finding task[1] and somewhat useful to ad hoc retrieval by result combination[2]. In this year, our conclusion was consistent with formers. Besides, the use of the URL and links inside the webpage were also observed. Again, results on training set are encouraging.

We made an assumption that a key resource is more likely to link to multiple relevant documents. Then the out-degree of the page and the similarities of the documents the page point to were used as the two factors for key resource selection. Experimental results were quite good, showing their ability of finding key resource on one server.

Two site uniting (SU) approaches have been studied to select proper pages as the representation of one server. (1) The document which has index characteristic and has a high enough similarity is reserved as key resource. (2) Documents of the same server in result list are given different reliability factor which is decaying by decreases of similarities. Both are useful for given examples (using as training set) in this year's Web track, especially the latter one. Better results were got by combing SU approach and out-degree factor mentioned above to find key resource.

All the experiments we performed were run on Okapi system. There are quite a few parameters to tune, which affect the performance greatly. Therefore, we also proposed and implemented a genetic algorithm based dynamic parameter learning approach to all the tasks.

2 Data preprocessing

2.1 Word and document pruning

Odd characters are meaningless to users and may bring on exception in processing. We cleaned characters that are unprinted and unrelated with formats. Also, the words containing more than 20 characters were deleted as they were deemed to incorrect words that affect collection statistics.

As we discovered that two bunches of documents had no content, we pruned them. One bunch was the set of files with postfix of "jpg" or "gif' (totally 151 files). The other bunch was most redirect html documents (17,086 files) as they only acted as gangways to destined documents that contained detailed content. The exceptions were those documents that redirected to themselves in order to refresh periodically. In all, we removed 17,235 documents (note there's documents overlap between two bunches). No relevant documents were lost.

2.2 Html parsing

As non-html pages (doc, ps and pdf) were much longer than html pages and had no html tags, we divided the collection into two, non-html (153,775 files) and html (1,076,743 files), and indexed

* Supported by the Chinese National Key Foundation Research & Development Plan (Grant G1998030509), Natural Science Foundation No.60223004, and National 863 High Technology Project No. 2001AA114082.

these two subsets respectively.

Html pages were parsed for two goals. One is to convert the file to XML format. For example, text in the tags of <b>, <strong> and <u> was extracted and marked with a new XML tag <srhB>. It was preparation for using html structure to improve retrieval. The other goal is to remove invisible text, such as comment text and codes in scripts, because they are meaningless to users.

3 Using of document structure

HTML document structure is studied in our experiments. We found that using keywords, bold text and title fields of the in-link pages do help on named page finding task. We use these three parts of the in-link webpage and in-link anchor text to build a new document of current page, and then index and retrieve on this new dataset. Although result of the new dataset is not good, by combing this result and result on original dataset, we got some improvement, shown in Table 3.1.

Table 3.1 Effects of using document structure on named page finding task

Method	Content based retrieval	Anchor + special fields	Combined
MRR	0.690	0.530	0.717

To topic distillation task, fields of bold font (<B>) and keywords in <meta> are also useful to the retrieval. It does help by giving a different term weight from the other full text.

4 Using link structure

4.1 Re-Ranking based on link analysis

Intuitively, counting the links to a document has been used to estimate the document's quality. However, the concept of key resource is different from the concept of quality. Therefore, we used some other features, such as Kleinberg's hub score, Kleinberg's authority score + hub score and out degree, to estimate whether the document is a key resource. The experiment on the training examples showed some improvement (see Table 4.1), but the result was disappointing to 50 topics of web track. The training set included seven topics in track guidelines.

Table 4.1 Finding key resources with link analysis on training topics

	Baseline	By linke analysis based re-ranking
Average precision of top 20 results	0.3827	0.4395

4.2 Site Uniting

The definition of key resource implied that it was most possible that only one page was key resource among pages from an identical site. As the list of content-based retrieval contains all the relevant pages, some might be ranked adjacently. Even worse, all of them were ranked high and thus pressed a possible key resource from another site lower. Therefore, we re-ranked the list by enhancing the page with highest rank from each site. The approach is called *Site Uniting* (SU). The algorithm can be shown as following, where F_1=1.03, F_2=1.01, F_3=1.005 in our experiments:

1) Divide the list to sub-lists, all pages in one sub-list come from an identical site.
2) To one sub-list, give the first, second and third highest similarity the weight F_1, F_2 and F_3, respectively. 4.1
3) Merge all the sub-lists into one and re-rankit.
The additional condition is that $F_1 > F_2 > F_3$.

The p@10 was lower than the base although 11-point precision was a little higher.

5 Using URL

In topic distillation task, the URL is also used to the retrieval. There are two functions can be provided by URL: (1) searching and shrinking; (2) scoring and selecting. On searching and shrinking, we give a "right level" to the return results within a server. The shrinking is based on three principles: i. Pages with more keywords matched in the URL is more important. ii. The location of the match effects the importance of the page, the righter the better; iii. To pages with the same conditions i and ii, shorter URL is better. On scoring and selecting, a keyword search is performed on URLs and got a result list which is useful to re-rank content-based retrieval result.

On named page finding tasks, we tried the URL classification. Using URL types (TNO-UTwente TREC 10 report [3]) proved to be a success in Entry Page Finding task last year. Unfortunately, it doesn't help this year. We analyzed 100 of the 150 correct answers. Table 5.1 shows how Named Pages distribute over the 4 URL types (root, sub root, path and file). Compared with Table 5.2, we conclude that Named Pages have almost the same distribution over this kind of URL classification as ordinary web pages. That means URL type is not a useful character for this year's task.

Table 5.1 URL type distribution in qrels

URL type	#page	percent
Root	2	2%
Sub root	1	1%
Path	6	6%
file	91	91%

Table 5.2 URL type distribution in corpus

URL type	#page	percent
Root	11680	0.6%
Subroot	37959	2.2%
Path	83734	4.9%
file	1557719	92.1%

6 Combination of distributed Retrieval results

As described in former section, the corpus has been divided into two data sets: one is for html document called html database, another one is for the remaining, call extra database. Retrieval has been done separately on these two distributed databases. How to combine the two result list is one of the interesting issues. The algorithm we used is to find a start rank point in html results, and insert the extra results from this point with some interval. The selection formula of start-point is:

$$Start = (2 - Sim\_extra/Sim\_html) * k + b \qquad 6.1$$

where k and b are constants. In our experiments, k=150 b=14.

7 Unsupervised dynamic parameters learning

The similarity between queries and documents is computed by BM2500[4]. There are quite a few parameters to be set, such as b, k_1, k_3, $avdl$. Especially b and k_1 play an important role in the performance. Parameters by training data, however, are always not suitable and helpful when dataset or queries change. At the same time, relevance judgments are not available while retrieving, thus supervised learning algorithms do not help. In this section, an unsupervised dynamic parameters (b and k_1) learning algorithm is described.

We use Genetic Algorithm (GA) for learning process. The fitness function in GA determines whether each set of parameters is good or not and the survival probability of each set[5]. According to the fact of lacking relevance judgment, it is required to find appropriate fitness functions which are oriented from the data and the retrieval themselves. The one we used is the summation of the similarity scores of top n relevant documents, shown in Eq(7.1).

$$fit\_fun_1 = \sum_{i=1}^{50}\sum_{j=1}^{1000} sim_{i,j} \qquad 7.1$$

Figure 7.1 and 7.2 show the correlation of using summation of similarities and the 11-point average precision, and the correlation of P@10 and summation of similarities on TREC10 and TREC2002 dataset, respectively.

Figure 7.1 Correlation between the two fitness functions in TREC2001 data

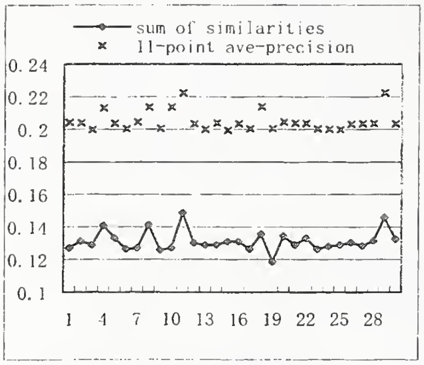
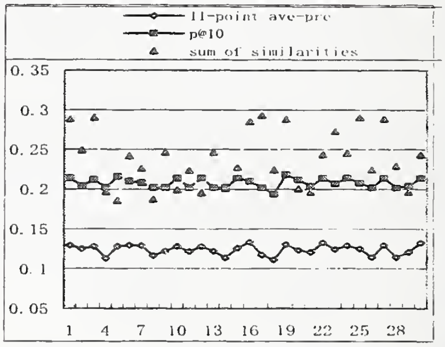


Figure 7.2 Correlation between three fitness functions in TREC2002 data



8 Runs Submitted and Evaluation Result

Table 8.1 Runs submitted on Topic distillation task

Run	Description	P@10
Thutd1	Thutd4 + outdegree	19.80%
Thutd2	Thutd4 + anchor re-ranking	22.45%
Thutd3	Thutd4 + site uniting	23.06%
Thutd4	Thutd5 + anchor combination + extra db result	21.43%
Thutd5	On Html db, long query	25.10%

Table 8.2 Runs submitted on Named page finding task

Run	Description	MRR
Thunp1	Content method	0.690
Thunp2	Combining inverse rank of Content and special fields results	0.530
Thunp3	Thunp5 + URL hierarchy	0.719
Thunp4	Thunp1 + URL hierarchy	0.687
Thunp5	Re-ranking of content and special fields results	0.717

Reference

[1] Craswell. N.. etc.. Effective site finding using link anchor information. In *SIGIR-01*.

[2] Gaojianfeng. etc.. TREC-10 Web Track Experiments at MSRCN, in TREC-10.

[3] Thijs Westerveld. etc.. Retrieving web pages using content. links. URLs and anchors. in TREC-10.

[4] Robertson, S. E., and Walker, S., Okapi/Keenbow at TREC-8. In *TREC-8*.

[5] Y. S. Chen. C. Shahabi. Automatically Improving the Accuracy of User Profiles with Genetic Algorithm. International Conference on Artificial Intelligence and Soft Computing, 2001.

University of Alicante Experiments at TREC-2002[1]

Vicedo, Jose Luis & Llopis, Fernando & Ferrández, Antonio

{vicedo,llopis,antonio}@dlsi.ua.es

Dpto. Lenguajes y Sistemas Informáticos
Universidad de Alicante
Apartado 99. 03080 Alicante, Spain

Abstract

This paper describes the architecture, operation and results obtained with the Question Answering prototype developed in the Department of Language Processing and Information Systems at the University of Alicante. This system is based on our TREC-10 approach where different improvements have been introduced. Main modifications reside on the introduction of a filtering stage into paragraph selection and answer extraction modules that allow the treatment of questions with no answer in the document collection. Moreover, WordNet has been enhanced by adding a collection of gazetteers that includes several types of proper nouns (people, organisations, and places) and a large variety of acronyms, measure and money units.

1. Introduction

This year, question-answering task has been significantly modified. The organisation has restricted the proposed experiments to *main* and *list* tasks. Main task is similar to previous year task but instead of permitting 5 ranked responses for each query and a maximum of 50 bytes as answer length, only a response is allowed and the answer string must contain nothing other than the exact answer. Besides, there is no guarantee that an answer will actually appear in the document collection. The list task consists of answering questions that will specify a number of instances to be retrieved. In this case, it is guaranteed that the collection contains at least as many instances as the question asks for.

The system presented to TREC-2002 QA task departs from the system presented in past TREC conferences [7][8] where new tools have been added and existing ones have been updated and adapted to cope with new specifications. Main enhancements rely on several aspects. First, passage selection and answer extraction stages have been adapted in order to face questions with no answer in the document collection. For this purpose, these stages have been complemented with a filtering module that rejects relevant paragraphs as well as possible answers that do not validate a series of restrictions. This way, when no possible answer remains after applying these restrictions, the system returns NIL as final answer. Second, WordNet has been extended by adding entities included in several gazetteers mainly referring to places (countries, states, cities, etc.) as well as and a large number of different acronyms, measure and money units. In this case, WordNet enrichment tries to minimise, as possible, the lack of a Name-Entity tagger.

Although our participation has been restricted to main task, this year we tried to face up all the specifications. In fact, it is the first time we manage with no-answer questions.

[1] This work has been partially supported by the Spanish Government (CICYT) with grant TIC2000-0664-C02-02 and (PROFIT) with grant FIT-150500-2002-416.

This paper is structured as follows: Section 2 describes system structure and operation and tries to emphasize new contributions. Afterwards, we present and analyse the results achieved and finally, we extract initial conclusions and discuss directions for future work.

2. System Overview

Our QA system is structured into four main modules: *question analysis, document/passage retrieval, paragraph selection* and *answer extraction.* First module processes questions expressed in open-domain natural language in order to analyse the information requested in the queries. This information is used as input by remaining modules. Document retrieval module accomplishes a first selection of relevant passages by using a passage retrieval system. Afterwards, the paragraph selection module filters these passages in order to select smaller text fragments (paragraphs) that are more likely to contain the correct answer. Finally, the answer selection module processes these fragments in order to locate and extract the final answer. Figure 1 shows system architecture.

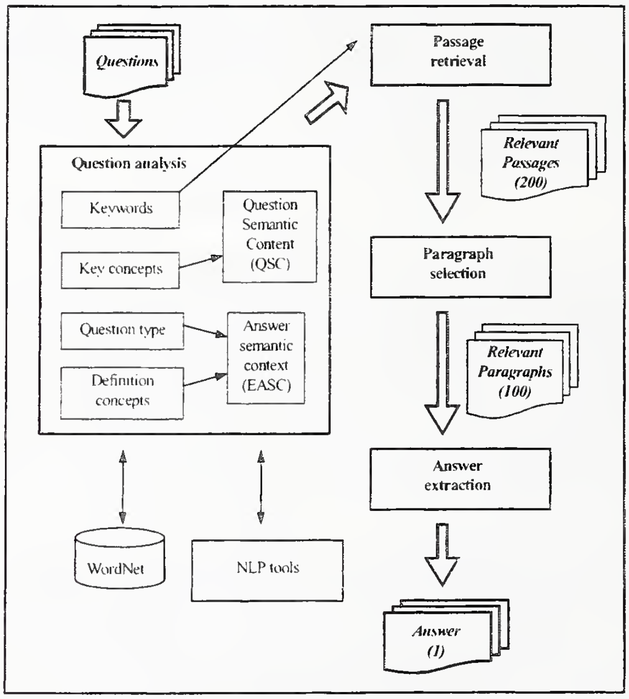


Figure 1. System architecture

2.1. Question Analysis

Question processing module accomplishes several tasks. First, questions are part-of-speech tagged and parsed. This process allows identifying simple noun and verbal phrases (*concepts*) in the query. Afterwards, this module determines *question type* and classifies concepts into two categories: *key* or *definition* concepts. Finally, these concepts are processed to obtain and represent its semantic characteristics. The resulting structures will be used as main information units for QA purposes.

Question analysis process starts with question type detection. This process maps Wh-terms (What, Which, How, etc) into one or several of the categories listed in figure 2. When no category can be detected by Wh-term analysis, NONE is used (e.g. "What" questions). This module also includes *definition* as a question type. Definition questions are detected by applying pattern-matching techniques.

Group A :	PERSON	LOCATION	GROUP	TIME	QUANTITY	NONE
Group B :	REASON	MANNER	DEFINITION			

Figure 2. Question type categories

Question type categories pertaining to *group A* are related to WordNet top concepts [2]. Each of these categories is represented by the vector of WordNet synsets that are semantically related to its corresponding top concept (called *QTC-Question type characteristics*). These synsets are obtained by extracting from WordNet all hyponyms of each top concept until third level and they are weighted depending on its level into the WordNet hierarchy and the frequency of its appearance into the path. As it can be deduced, NONE questions have an empty QTC.

Once question type has been obtained, the system selects the noun phrase in the query that expresses the semantic characteristics of the expected answer (*definition concept).* Definition concepts do not help the system to locate the correct answer into the document collection but they usually add critical information about the kind of information requested by the query. The semantic characteristics of definition concepts are represented by a weighted vector (*CT- concept type*) that includes the set of synsets they are semantically related to. These synsets are obtained by extracting from WordNet all hyperonyms of each definition concept head term (its path to top concepts) and they are weighted depending on its level into the WordNet hierarchy and the frequency of its appearance into the path towards top concepts.

QTC and definition concept CT are used to generate the *expected answer semantic context* (EASC). This context defines the semantic context that the expected answer has to be compatible with. This context is computed by performing exclusive vectorial addition between QTC and CT. As special case, EASC will be equal to CT for NONE questions. By the other hand, *group B* questions have a different treatment due to the special nature of its expected answers and therefore, at this analysis point only question type assignment is needed.

Once definition concepts have been detected, remaining question concepts are classified as *key concepts*. This question processing stage builds the semantic representation of the *key concepts* expressed into the query (*semantic content of a question - QSC*). This process consists of obtaining a general semantic representation of the concepts that appear in the question.

The head of a *key concept* syntactic structure represents the basic element or idea the concept refers to. Remaining terms pertaining to this structure modify this basic concept by refining the meaning represented by its head. Following this approach, the system tries to obtain and represent the different ways of expressing a concept. This process starts by associating each term pertaining to a concept, with its synonyms and one level search hyponyms and hyperonyms. These relations are extracted from WordNet lexical database. We define the semantic content of a term *t* (SC*t*) as a set of terms made up by the term *t* and all the terms related with it through the synonym and one level search hyponym and hyperonym relations. The SC of a term is represented using a weighted term vector. The weight assigned to each term pertaining to the SC of a term *t* is the 80%, 50% and

50% of the *idf* [3] value of term *t* for synonyms, hyponyms and hyperonyms respectively. As a concept is made up by the terms included into the same syntactic structure, we define the semantic content of a concept (SCC) as the set of weighted vectors (HSC, MSC) were HSC is the a vector obtained by adding the SC of the terms that made up the head of the concept and MSC is the vector resulting from adding the SC of terms that modify that head into the same syntactic structure. The set of SCCs that stand for the concepts appearing in a question builds the semantic content of a question (QSC). This way, the QSC represent all the concepts referenced into the question and the different ways of expressing each of them. All the described processes and related formulae are widely described and explained in [6].

Figure 3 sums up these processes using an example question. First, the system identifies and classifies the concepts *"company", "manufacture"* and *"American Girl doll collection"* by parsing question. Afterwards, the system generates the expected answer semantic context (EASC) and obtains the semantic content of each key concept to compound semantic content of the question (QSC).

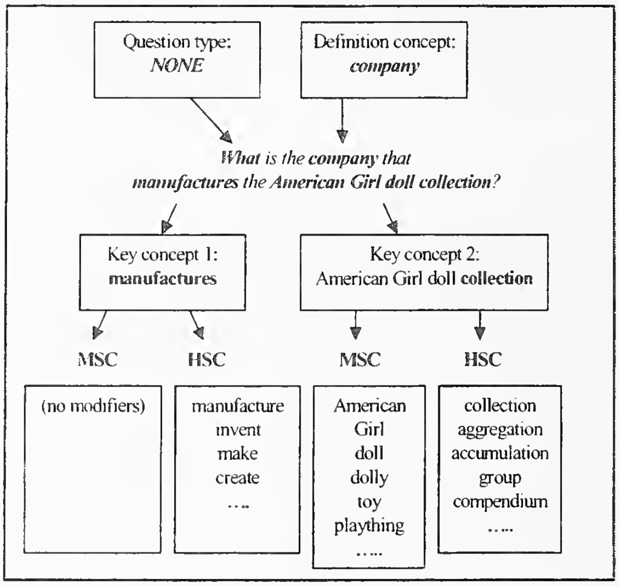

Question type: *NONE*
Definition concept: *company*
What is the company that manufactures the American Girl doll collection?
Key concept 1: manufactures
Key concept 2: American Girl doll collection
MSC
HSC
MSC
HSC
(no modifiers)
manufacture invent make create ....
American Girl doll dolly toy plaything .....
collection aggregation accumulation group compendium .....

Figure 3. Question analysis processes

Question keywords are used for first stage passage retrieval, QSC information will help paragraph selection module to detect the paragraphs that are more likely to contain the answer and finally, EASC (or question type for group B questions) will allow detecting and evaluating possible answers.

2.2. Passage retrieval module

First stage retrieval applies the passage retrieval approach described in [1]. This passage retrieval can be applied over all the document collection, but it has only been applied for thc 1.000 relevant documents supplied by TREC organisation. Therefore, keywords detected at question processing stage are used for retrieving the 200 most relevant passages from the documents included in this initial list. This process is intended to reduce the amount of text that has to be processed by NLP modules since these passages are made up by text snippets of 15 sentences length.

2.3. Paragraph selection

This module processes the 200 first ranked passages selected at passage retrieval stage in order to extract smaller text fragments that are more likely to contain the answer to the query. As all this process is widely described in [7][9] we extract here the basic algorithm:

1. Documents are split into sentences.
2. Overlapping paragraphs of three sentences length are obtained.
3. Each paragraph is scored. This value measures the similarity between each paragraph and the question.
4. Paragraphs are ranked according to this score.

The score assigned to each paragraph (*paragraph-score*) is computed as follows:

a) Each SCC appearing in the question is compared with all the syntactic structures of the same type (noun or verbal phrases) appearing into each relevant paragraph. Each comparison generates a value. As result, each SCC is scored with the maximum value obtained for all the comparisons accomplished through the paragraph.

b) The paragraph-score assigned to each paragraph is obtained by adding the values obtained for all SCCs of the question as defined in previous step.

c) The value that measures similarity between a SCC and a syntactic structure of the same type is obtained by adding the weights of terms appearing into SCC vectors and the syntactic structure that is being analysed. If the head of this syntactic structure does not appear into the vector representing the SCC head (HSC), this value will be 0 (even if there are matching terms into MSC vector).

2.3.1. Filtering relevant paragraphs

This module has been added with the intention of managing with questions with no answer. This module filters relevant paragraphs by getting rid of those that contain value 0 for more than one SCC evaluated previously. This way, the system only accepts, as relevant, a paragraph that contains nearly all key concepts expressed in the query. At this stage, best 100 ranked paragraphs are selected to continue with the remaining processes.

2.4. Answer extraction

This process consists on analysing selected paragraphs in order to detect and evaluate concepts that can be considered probable answers. Among all the candidates the system will select the one it considers the correct answer. Answer extraction processes differ depending question type group.

For *group A* questions, the system gets rid of key concepts appearing in the paragraph and selects the concepts that validate lexical restrictions of the expected question (e.g. proper noun for a Who question). All these concepts are considered probable answers of the query. Next, the system computes the semantic context of each possible answer (SCPA) by taking into account the semantic concept types (CT) of the probable answer and its adjacent concepts in the paragraph. This way, The SCPA of a probable answer *r* is computed as:

$$SCPA_r = CT_{(r-1)} + CT_r + CT_{(r+1)}$$

Then, probable answers are filtered and only those that are compatible with the expected answer semantic context (EASC) are selected. For this purpose, each probable answer is assigned a score (*probable-answer-compatibility*) that measures its compatibility with the expected answer semantic context. Only probable answers with score greater than 0 are maintained. This value is computed as follows:

$$\textit{probable-answer-compatibility}_r = \cos(EASC, SCPA_r)$$

Next, compatible probable answers are evaluated by computing a final score (*answer-score*) that is obtained as follows:

$$\textit{answer-score}_r = \textit{paragraph-score} \cdot \textit{probable-answer-compatibility}_r$$

Intuitively, the *answer-score* combines (1) the semantic compatibility between the probable answer and the expected answer (*probable-answer-compatibility*) and (2) the degree of similarity between question and paragraphs (*paragraph-score*). Finally, probable answers are ranked on *answer-score* and the system returns the first one as correct answer or NIL when no compatible probable answers have been found.

Answer extraction manages differently with *group B* questions (*definition, reason* and *manner*). The answer to this kind of questions is usually a part of a sentence that defines a concept, reason or a way of performing an action and they are usually expressed via certain sentence syntactic structures. Consequently, our approach performs probable answer detection and extraction by applying syntactic pattern-matching techniques over relevant paragraphs. This way, when no pattern has been successfully validated, the system returns NIL as answer. This approach, as well as a full description of the patterns is described in [6].

3. Results

We submitted one single run for main task. This task allowed one answer for each question and the response had to contain only the exact answer string to be considered correct. Figure 4 shows the results obtained.

Number wrong:	302	
Number unsupported:	2	
Number inexact:	15	
Number right:	181	
Confidence-weighted score:	0.496	
Precision of recognizing no answer:		39 / 250 = 0.156
Recall of recognizing no answer:		39 / 46 = 0.848

Figure 4. TREC-2002 results

Our main objective was to inspect the way that restrictions imposed at paragraph selection and answer extraction stages affected system performance as a whole and, particularly, to the treatment of no-answer questions.

Result analysis shows two main circumstances to take into account. First, system performance presents a good precision since it has answered correctly a 72.4% of the questions the system considered to have answer in the collection (181 from 250 not NIL answers). Nevertheless, these filters seem to be too restrictive since the system has provided a NIL response for 211 questions with known answer. Second, our filtering approach does not perform correctly the detection of no answer questions. In fact, the precision achieved in this task has been very low (only a 15.6%). Moreover, despite of having answered as NIL a large number of questions (250), seven real NIL questions have not been recognised (a 15.2% of the 46 existing NIL questions).

Comparison with TREC-9 and TREC-10 results.

Comparison between our different participations is difficult and has to be analysed carefully since task specifications are significantly different. Nevertheless, we can compare 50-bytes strict results achieved in previous conferences with TREC-2002 results if we focus our attention mainly on the percentage of correct answers ranked in first place (see third column in figure 5). From this point of view, our system has achieved a significant improvement in precision since the percentage of correct answers retrieved in first place increases 12,8 points from TREC-10 results.

	% Answers found	% Answers in 1st place
TREC-9	33,9%	16,9%
TREC-10	39,6%	23,4%
TREC-2002	36,2%	36.2%

Figure 5. TREC Participation results

4. Future Work

As it can be deduced from result analysis, the main objective pursued this year has not been achieved. The filtering processes incorporated to paragraph selection and answer extraction stages have significantly increased system precision, they have failed in detecting questions with no answer in the collection. Consequently, we need to direct our next steps to investigate and test validation techniques that could cope efficiently with no answer questions.

5. References

1. Llopis F. and Vicedo J.L. *IR-n: a passage retrieval system at CLEF-2001*. In proceedings of the second Cross-Language Evaluation Forum (CLEF2001). Lecture Notes in Computer Science. September 2001. Darmstadt (Germany).
2. Miller G.(1995), *"Wordnet: A Lexical Database for English"*, Communications of the ACM 38(11) pp 39-41.
3. Salton G.(1989). *Automatic Text Processing: The Transformation, Analysis, and Retrieval of*

Information by Computer. Addison Wesley Publishing, New York.

4. TREC-9, 2000. Call for participation Text Retrieval Conference 2000 (TREC-9).
5. TREC-10, 2001. Call for participation Text Retrieval Conference 2001 (TREC-10).
6. Vicedo J.L. *SEMQA: Un modelo semántico aplicado a los sistemas de Búsqueda de Respuestas*. Phd Thesis. May 2002.
7. Vicedo J.L., Ferrandez A. And Llopis F. *University of Alicante at TREC-10*. In Proceedings of the Tenth Text Retrieval Conference. November 2001. Gaithersburg (USA).
8. Vicedo J.L. and Ferrandez A. *A semantic approach to Question Answering systems*. In Proceedings of the Ninth Text Retrieval Conference. November 2000. Gaithersburg (USA).
9. Vicedo J.L. *Using semantics for Paragraph selection in Question Answering systems*. In proceedings of the Proceedings of the Eighth String Processing and Information Retrieval Conference (SPIRE'2001). November 2001. Laguna de San Rafael (Chile).

The University of Amsterdam at TREC 2002

Christof Monz   Jaap Kamps   Maarten de Rijke
Language & Inference Technology group
ILLC, University of Amsterdam
Nieuwe Achtergracht 166, 1018 WV Amsterdam
The Netherlands
E-mail: {christof,kamps,mdr}@science.uva.nl
URL: www.science.uva.nl/~{christof,kamps,mdr}

Abstract: We describe our participation in the TREC 2002 Novelty, Question answering, and Web tracks. We provide a detailed account of the ideas underlying our approaches to these tasks. All our runs used the FlexIR information retrieval system.

1 Introduction

At TREC 2002 we took part in the Novelty, Question Answering, and Web tracks. Our main aims for the Novelty and Web tracks was to set up baseline systems on which we plan to build in future editions of the tracks. Our main aim for the Question Answering track was to test a revised architecture of our knowledge-intensive question answering system Tequesta [16], and to experiment with a number of newly added features relating to the document retrieval steps carried out within Tequesta.

For all three tracks, our experiments exploited the FlexIR information retrieval system developed at the University of Amsterdam [15]. The main goal underlying FlexIR's design is to facilitate flexible experimentation with a wide variety of retrieval components and techniques. FlexIR is implemented in Perl, and built around the standard UNIX pipeline architecture; it supports many types of pre-processing, scoring, indexing, and term-weighting methods, of which we made good use this year. Depending on the task at hand, we used different weighting schemes; see the detailed descriptions of our efforts for each of the tracks below for the exact settings.

The rest of this paper is organized as follows. In three (largely self-contained) sections we describe our work for the Novelty, Question Answering, and Web tracks. We also provide a brief concluding section.

2 Novelty Track

In this section we describe our submissions for the TREC 2002 novelty track. The overall aim of the track is to investigate systems' abilities to locate relevant *and* new information within the ranked set of documents retrieved in a reply to a search engine query. Thus, systems should return information that is both new and relevant rather than whole documents containing duplicate and extraneous information [8]. The novelty task can naturally be divided into two parts. Indeed, the guidelines require that participants identify two lists of documents for a given topic [20]. The first contains the *relevant* sentences, and the second one (a subset of the first) contains only those sentences that add *new* information.

Our main interest in participating in the novelty track was in exploring the second part of the task: identifying *new* sentences. However, due to time constraints we had to limit ourselves to fairly straightforward approaches to both parts of the novelty task. We ended up setting a simple baseline, using established IR strategies for the relevance part, and weighted overlap for the novelty part; our aim is to build on this with more linguistically motivated techniques in the near future. The relevance part, which is the most important part of the track as it also has an obvious impact on the performance of the novelty part, requires far more work than we had anticipated.

The remainder of this section is organized as follows. After recalling some key facts about the experimental set-up, we describe our approaches to the relevance and novelty parts of the novelty task, and then list and briefly discuss our results.

2.1 Topics and Documents

For ease of reference, we briefly highlight some key facts about the documents and topics used in the novelty track; the overview paper provides further details [8]. Initially, there were 50 topics, taken from TRECs 6, 7, and 8 (topics 300–450); after the evaluation was completed, one topic was removed as it was not found to have relevant sentences. The documents are a subset of the relevant documents for the topics. Participants are provided with a ranked list of relevant documents, with between 10 and 25 relevant documents per topic.

2.2 Computing Relevance

We approached the task of identifying *relevant* sentences in the following manner. For a given topic, the sentences in the relevant documents for that topic were viewed as documents themselves, thus creating a sentences-as-documents collection for each topic. We ran the topic (only using the title and description fields) against this sentences-as-documents collection using our retrieval engine FlexIR. We initially followed Salton and Buckley, who recommend the tfx.nfx weighting scheme for short queries and short documents [18], but some informal pre-submision experiments on comparable topics and documents suggested that tfv.nfx was somewhat more effective.

Three different runs were submitted: one where all documents and topics were porter stemmed [17] (run identifier `UAmsT11ntste`), and a second where they were lemmatized using Helmut Schmidt's TreeTagger [19] (run identifier `UAmsT11ntlem`); here, each word is assigned its syntactic root through lexical look-up; mainly number, case, and tense information is removed, leaving other morphological processes such as nominalization intact. And in the third run the results of the other two runs were simply merged (run identifier `UAmsT11ntcom`). Our motivation for the first two runs was to see to which extent morphological normalization has an impact on the relevance and novelty parts of the task. The third run was included to determine the impact on the novelty part of the task of high recall approaches to the relevance part.

2.3 Computing Novelty

Our approach to the novelty part of the task was based on a non-symmetric weighted overlap score, which we use to provide graded answers to the following question: is the information contained in a sentence *entailed* by a sentence (or set of sentences) seen before? We say that a sentence is *new* (within a context) if it is not entailed by the context.

Assuming the usual definition of *idf* term weights, we compute the *entailment score*, $entscore(s_i, s_j)$, of two (sets of) sentences s_i and s_j by comparing the sum of the weights of terms that appear in both s_i and s_j to the sum of the weights of all terms in the second sentence (or set of sentences) s_j:

$$entscore(s_i, s_j) = \frac{\sum_{t_k \in (s_i \cap s_j)} idf_k}{\sum_{t_k \in s_j} idf_k}. \tag{1}$$

In words: how many of the content-bearing terms in s_j occur in s_i? Clearly, $entscore(s_i, s_j)$ varies from 0 to 1.

A few remarks are in order. First, note that our entailment score is not just a notion of similarity: in general, $entscore(s_i, s_j) \neq entscore(s_j, s_i)$.

Second, to work with *entscore* and conclude that s_i entails s_j, it may not be sufficient to have a non-zero entailment score: we may need some positive 'entailment threshold.' In our experiments we used 0.6; this figure was obtained by testing our methods on the 4 samples provided by NIST as training material. The mechanism of entailment thresholds offers a large amount of flexibility for fine-tuning the entailment notion to one's purposes; see below for some discussion on this point.

To identify the list of new sentences as required by the guidelines, we simply went down our list of relevant sentences, taking the first one as our starting point, and including later ones only if they were not entailed by the ones already included. Our three runs used exactly the same ideas for their novelty parts, and differed only in the list of relevant sentences they took as input.

2.4 Results and Discussion

To assess the results of the relevance and novelty parts of the task, the product of precision and recall (P*R) is used as measure, with separate scores for the two parts of the task. The average of P*R is meaningful even when the judgment sets sizes vary widely, as is the case for the task at hand. One downside of P*R is that in practice the scores tend to be close to 0.

Table 1 shows the results for each of our three runs. Taking the stemmed run as our baseline, we see that both lemmatizing and combining produce significant improvements, for both the relevance and novelty parts.

Table 1: Summary of the results for the novelty track.

	Average P*R	
Run identifier	Relevance	Novelty
`UAmsT11ntste`	0.029	0.028
`UAmsT11ntlem`	0.033 (+13.8%)	0.031 (+10.7%)
`UAmsT11ntcom`	0.034 (+17.2%)	0.032 (+14.3%)

Let's take a closer look at the results. The improvements obtained by lemmatizing topics and documents instead of stemming them, are not uniform. For many individual topics stemming is at least as good as, or even better than lemmatizing, for both relevance and novelty; similar observations can be made about the combined run vs. the other runs. Table 2 provides a breakdown of the number of top scores per run; the first number is the total number of top scores for a given run, the second number is the number of unique top scores (that are not shared by other runs).

Table 2: Top scores per run.

	# Top P*R Scores (shared, unique)	
Run identifier	Relevance	Novelty
`UAmsT11ntste`	25, 3	23, 12
`UAmsT11ntlem`	37, 15	37, 26
`UAmsT11ntcom`	21, 9	23, 0

Figures 1 and 2 plot our P*R scores against the median by topic. They suggest a number of things. First, while we seem to do relatively poorly on the relevance part of the novelty task, our performance on the novelty seems somewhat better.

The definition of the novelty task suggests that a system's performance on the novelty part is, to a large degree, determined by its performance on the relevance part, and the considerable similarity between the plots in Figures 1 and 2 confirms this.

We carried out a number of post-submission experiments, using the golden standards provided by NIST. First of all, we

Figure 1: Comparison of relevance scores to median by topic.

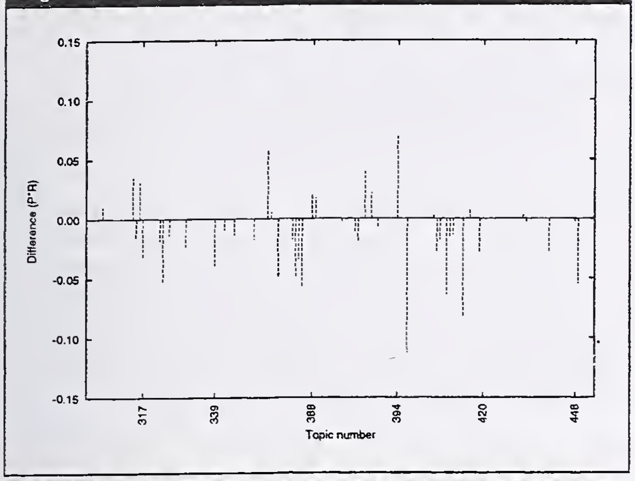
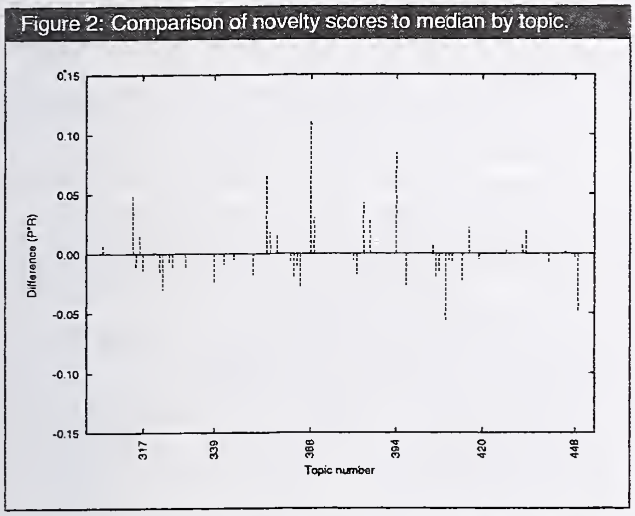


Figure 2: Comparison of novelty scores to median by topic.



ran some experiments to see whether we used an (almost)optimal value for the entailment threshold for our official submissions. Figure 3 shows the average precision, recall, and P*R scores for our combined run (UAmsT11ntcom) with increasing values of the threshold. The value of 0.6 that we used in the submitted run is close to the optimal one, although values of 0.7 or higher would have produced slightly higher scores (0.033, +3.1%).

Figure 3: Impact of the entailment threshold on novelty.

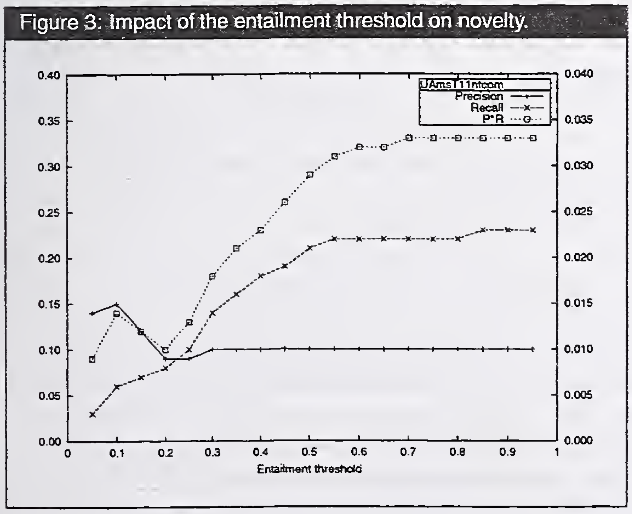
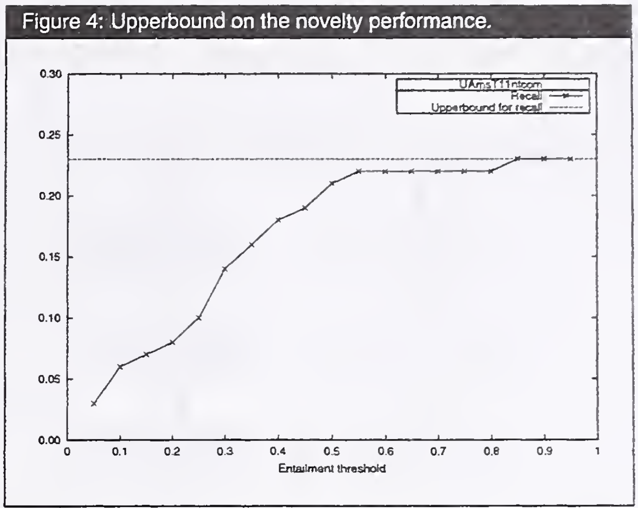


Figure 4: Upperbound on the novelty performance.



Furthermore, we determined an upperbound on the performance of the novelty part of our system, to get some understanding of its behavior in absolute terms. If we take the relevance results of our best run (UAmsT11ntcom) and intersect these with the novelty qrels provided by NIST, we get the best possible list of new sentences (given our relevance output). Since the precision for this optimal list is 1, it only makes sense to look at the recall for this list, which turns out to be 0.23, very close to the score actually obtained (0.22); see Figure 4.

In conclusion, while we are especially interested in the novelty part of the novelty track, it seems that the relevance part is the hardest and most important part of the task. We plan to address it more extensively than we have done so far by bringing in linguistic features; it is not obvious, however, how much this will differ from document summarization.

3 Question Answering Track

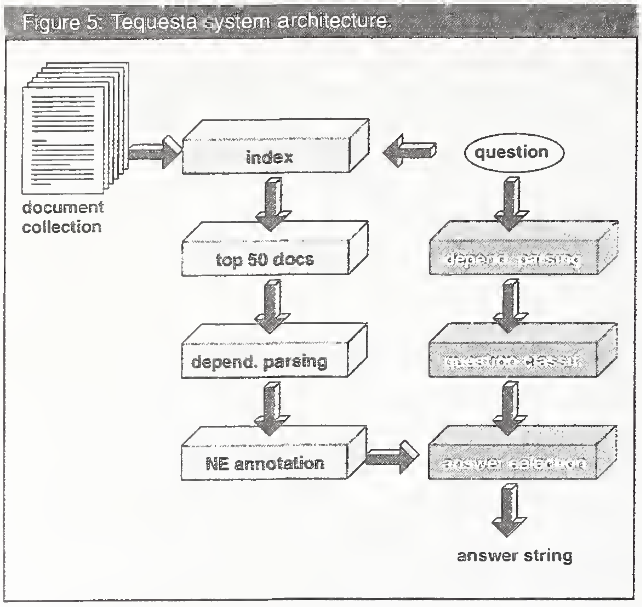
This section describes our submissions for the question answering track at TREC 2002. Our main focus was on evaluating a basic question answering system that exploits shallow NLP techniques in combination with standard retrieval techniques.

3.1 System Description

The system architecture of Tequesta (TExtual QUESTion Answering) is fairly standard; its overall architecture is displayed in Figure 5. Like most current QA systems, Tequesta is built on top of a retrieval system. The first step is to build an index for the document collection, in this case the AQUAINT collection. Then the question is translated into a retrieval query which is sent to the retrieval system. For retrieval we use the FlexIR system described in the introduction.

The retrieval system is used to identify a set of documents that are likely to contain the answer to a question posed to

Figure 5: Tequesta system architecture.



the system. The top documents returned by FlexIR are further processed as described in Section 3.1.2.

Just like the top documents, the question is also parsed. The parsed output is used to determine the focus of the question. Question analysis is explained in Section 3.1.3.

3.1.1 Document Retrieval

For pre-fetching relevant documents that are likely to contain the answer, Tequesta uses FlexIR, which was given a total of 1,033,461 documents to index. All our official runs for TREC 2002 used the Lnu.ltc weighting scheme [3] to compute the similarity between a question and a document. For the experiments on which we report in this article, we fixed *slope* at 0.2; the pivot was set to the average number of unique words occurring in the collection.

To increase precision, we decided to use a lemmatizer; the lemmatizer used is TreeTagger, the same as in our experiments for the novelty track.

In document retrieval it is common practice to return a ranked list of documents, each item being adorned with the similarity score. Additionally, FlexIR returns a minimally matching span (MSM) for each document. An MSM indicates the starting (s) and ending position (e) of a text excerpt, containing all matching terms, such that there are no positions s' or e', $s < s'$ and $e' < e$, and neither the span s', e nor the span s, e' also covers all matching terms; see also [4]. In a later stage of the question answering process, MSMs are used to restrict documents to passages which are likely to contain the answer.

3.1.2 Document Analysis

Document analysis focuses on the top 50 documents that were returned by FlexIR. For each of them, we used the MSM to extract a text passage which was then analyzed further. The passage begins with the sentence containing the beginning position of the MSM and ends with the sentence containing the ending position of the MSM. This way we make sure that the passage contains full sentences which can be parsed. Here, we used Dekang Lin's dependency parser MINIPAR [14]. Identifying sentence boundaries was accomplished by TreeTagger.

Depending on the question type — see below for more details — a named entity recognizer was applied to identify phrases that are of the same semantic type as the expected answer. This process is guided by the question classification component. For instance, if a question is looking for a numerical expression (such as age, speed, length, etc.) only expressions of that type are annotated.

3.1.3 Question Analysis

Just like the top 50 documents, the questions themselves were also part-of-speech tagged, morphologically normalized, and parsed. Since there is a significant difference between word order in questions and in declarative sentences, we needed to adjust the tagger for questions. To this end, TreeTagger was trained on a set of 500 questions with part-of-speech tags annotated. We used 300 questions taken from the Penn Treebank II data set together with the 200 TREC-8 questions, which we annotated semi-automatically.

We used 33 categories to classify the focus or target of a question, some of which are listed in Figure 6.

To identify the target of a question, pattern matching is applied to assign one of the 33 categories to the question. In total, a set of 102 patterns is used to accomplish this. Some of the patterns used are shown in Table 3.

If more than one pattern matches the question, it was assigned multiple targets. The patterns are ordered so that more specific patterns match first. Also, the answer selection component described in the next subsection obeys the order in which questions were categorized to find answers for more specific targets first.

Questions of type `what-np` form a special category. Here we use a dependency parser to identify the appropriate target, symbolized by np in the type. Usually, `what-np` questions are of the form *What NP VP?* or *What NP PP VP?*. After parsing the question, we use the head of the NP as target, which has *what*, or *which* as a determiner. For instance, question 1413 from the TREC 2002 question set, shown in (2), is assigned `what:river` as question target.

(2) What river is called "China's Sorrow"?

If none of the matching strategies described so far is able to assign a target to a question, the question is categorized as `unknown`. As a consequence, none of the answer selection strategies which are particularly suited for the respective question targets can be applied, and a general fall back strategy is used.

Table 3: Types for question classification.

<table>
<tr><th>Question target</th><th>Example patterns</th></tr>
<tr><td>name</td><td>/(W|w)hat(wa| i|\')s the name/</td></tr>
<tr><td>pers-def</td><td>/[Ww]ho(wa| i|\')s [A-Z][a-z]+/</td></tr>
<tr><td>thing-def</td><td>/[Ww]hat(wa| i|\')s an? /,/ (was|is|are|were) a kind of what/</td></tr>
<tr><td>pers-ident</td><td>/[Ww]ho(wa| i|\')s the/</td></tr>
<tr><td>thing-ident</td><td>/[Ww](hat|hich)(wa| i|\')s the /</td></tr>
<tr><td>number</td><td>/[Hh]ow (much|many) /</td></tr>
<tr><td>expand-abbr</td><td>/stand(s)? for(what)?\s*?/,/is (an|the) acronym/</td></tr>
<tr><td>find-abbr</td><td>/[Ww]hat(i|\')s (the|an) (acronym|abbreviation) for</td></tr>
<tr><td>agent</td><td>/[Ww]ho /,/ by whom[\.\?]/</td></tr>
<tr><td>object</td><td>/[Ww]hat (did|do|does) /</td></tr>
<tr><td>known-for</td><td>/[Ww]hy .+ famous/ /[Ww]hat made .+ famous/</td></tr>
<tr><td>aka</td><td>/[Ww]hat(i|\')s (another|different) name /</td></tr>
<tr><td>name-instance</td><td>/Name (a|one|some|an) /</td></tr>
<tr><td>location</td><td>/[Ww]here(\'s)? /,/ is near what /</td></tr>
<tr><td>date</td><td>/([Aa]bout)?(W|w)hen /,/([Aa]bout)?(W|w)(hat|hich) year /</td></tr>
<tr><td>reason</td><td>/[Ww]hy /</td></tr>
<tr><td>what-np</td><td>–</td></tr>
<tr><td>unknown</td><td>–</td></tr>
</table>

3.1.4 Answer Selection

Given the parsed and annotated top documents returned by FlexIR and given the parsed and classified questions, the actual process of identifying the answer starts.

Questions of type `agent` ask for an animate entity, such as a person or organization, being the logical agent of an event described in the question. If the dependency structure from the question matches a dependency structure from a document and there is an animate NP in subject position, or, in case of passive voice, within a PP headed by the preposition *by*, we take this to be the logical agent. Of course, such an NP is disregarded if it already occurs in the question itself. Questions of type `object` are dealt with analogously.

Questions of type `what-np` are particularly interesting because they are very frequent (at least in the TREC 2002 data, where 14.8% of the questions are of this type) and explicitly require some lexical knowledge base. Questions of type `what-np` ask for something that is an instance of the `np` and that fits the further description expressed in the remainder of the question. For example, question 1525, given in (3), asks for something which is a university.

(3) What university did Thomas Jefferson found?

In (3) *university* is the focus of the question and the further constraint *did Thomas Jefferson found?* is the topic of the question. In order to establish the relationship between an entity found in a matching dependency structure and the predicate *university* it is necessary to access a lexical knowledge base. Tequesta exploits WordNet for this purpose. In particular, WordNet's hyponym relations are used.

Answer candidates for all remaining question types where identified by named entity extraction where the named entity has to be of the same type as the expected answer.

Each answer candidate received a matching score depending on its position in the document. Candidates occurring within the MSM passage received a higher score than candidates occurring outside it. If the same candidate was extracted several times, possibly from different documents, their individual scores were summed up. The answer candidates were sorted by score and the answer candidate with the highest score was returned as answer. Answer candidates with identical scores were sorted randomly.

Since the score of the highest ranked answer candidate can be the sum of several occurrences, possibly from different documents, we take the document which has the largest share in the score as the supporting document, which is returned together with the answer-string.

3.1.5 Confidence

One of this year's changes in the TREC question answering track was to adorn an answer with a confidence score, indicating the system's trust in the returned answer. We used a rather simple approach to computing confidence. All answer candidates for a question q were ranked with respect to their answer score, yielding a sorted list of answer candidates $a_1, \ldots, a_n$, where $score(a_i) \geq score(a_{i+1})$, for $1 \leq i \leq n$. If two answer candidates have the same score, they are sorted at random. Then, the confidence that the highest ranked answer candidate is indeed the correct answer is computed as follows:

$$confidence(a_1) = \begin{cases} a_1 - a_2 & \text{if } a_1 > a_2 \\ \frac{1}{m} & \text{if } a_1 = \cdots = a_m > a_{m+1} \end{cases}$$

Figure 6: Question targets, plus examples from the TREC-11 question set.

`agent` name or description of an animate entity
(Q-1424): *Who won the Oscar for best actor in 1970?*

`aka` alternative name for some entity
(Q-1448): *What is the fear of lightning called?*

`capital` capital of a state or country
(Q-1520): *What is the capital of Kentucky?*

`date` date of an event
(Q-1406): *When did the story of Romeo and Juliet take place?*

`date-birth` date of birth of some person
(Q-1880): *When was King Louis XIV born?*

`date-death` date of death of some person
(Q-1601): *When did Einstein die?*

`expand-abbr` the full meaning of an abbreviation
(Q-1531): *What does NASDAQ stand for?*

`location` location of some entity
(Q-1818): *Where did Golda Meir grow up?*

`name` the name of a person or an entity in general.
(Q-1436): *What was the name of Stonewall Jackson's horse?*

`number-dist` spatial distance between two entities
(Q-1876): *How far from the earth is the sun?*

`number-height` height of some entity
(Q-1802): *How tall is Tom Cruise?*

`number-length` length of some entity
(Q-1857): *What is the length of Churchill Downs racetrack?*

`number-money` monetary value of some entity or event
(Q-1645): *How much is the international space stations expected to cost?*

`object` `object` questions are near-reverses of the `agent` questions. Here, the object of an action described in the question is sought.
(Q-1590): *What do grasshoppers eat?*

`pers-ident` a person fitting some description expressed in the question
(Q-1769): *Who is the owner of the St. Petersburg Times?*

`thing-ident` thing identical to the description expressed in the question
(Q-1547): *What is the atomic number of uranium?*

`what-np` an instance of the np fitting the description
(Q-1484): *What college did Allen Iverson attend?*

3.2 Results

The 2002 edition of the main QA task differs from previous years in several aspects. First of all, the document collection has changed from Disks 1–5 of the TIPSTER/TREC collection to the AQUAINT collection covering a more recent period, namely 1998–2000. A total of 500 questions is provided that seek short, fact-based answers. Some questions are not known to have an answer in the document collection. A further restriction, with respect to previous TRECs, is that each participating system is allowed to return only one response per question. A response is either a [answer-string, docid] pair or the string "NIL," The answer-string has to be an exact answer and the docid must be the id of a document in the collection that supports the answer.

An [answer-string, docid] pair is judged *correct* or *right* (R) if the answer-string consists of exactly a correct answer and that answer is supported by the document returned. If the answer-string is responsive and contains a correct answer, but the document does not support that answer, the pair will be judged "unsupported" (U). If the answer-string contains a correct answer and the document supports that answer, but the string contains more than just the answer (or is missing bits of the answer), it is judged as *inexact* (X). Otherwise, the pair is judged *incorrect* or *wrong* (W).

Finally, the scoring method for a run has changed in order to incorporate the confidence with which a question is answered by a system. Within the submission file the questions should be ordered from most confident response to least confident response. The final *confidence-weighted score* (CWS) is computed as follows:

$$\mathrm{CWS} = \frac{\sum_{i=1}^{500} \frac{1}{i} \sum_{j=1}^{i} [\![\, judgment(j) = \mathrm{R} \,]\!]}{500}$$

where $judgment(j)$ is the judgment of the NIST assessors for question j, and $[\![\, expression \,]\!]$ is 1 if *expression* is true, and 0 otherwise.

3.2.1 Submitted Runs

We submitted three runs for the main task (`UAmsT11qaM1`, `M2`, and `M3`).

The runs differed along 2 dimensions: the number of documents used as input for the answer selection process: either 50 documents (`UAmsT11qaM1`) or 100 documents (`UAmsT11qaM2` and `UAmsT11qaM3`), and whether questions were sorted with respect to confidence or not: runs `UAmsT11qaM1` and `UAmsT11qaM2` were sorted with respect to confidence and run `UAmsT11qaM3` was simply sorted by question id.

3.2.2 Results and Discussion

Table 4 summarizes the confidence-weighted scores (CWS) for each of our three submitted runs (`UAmsT11qaM1`, `M2`, and `M3`) over the 500 questions.

To investigate the impact of the different judgments for partial correctness of an answer-string, we compared the strict confidence-weighted scores, as defined above, to confidence

Table 4: Summary of the CWS for the main task.

UAmsT10qa...	M1	M2	M3
CWS(R)	0.145	0.101	0.146
CWS(R,U)	0.219	0.213	0.197
CWS(R,X)	0.151	0.135	0.174
CWS(R,U,X)	0.225	0.248	0.226

scores where also inexact (X) or unsupported (U) answers count as correct. E.g.,

$$\text{CWS(R,X)} = \frac{\sum_{i=1}^{500} \frac{1}{i} \sum_{j=1}^{i} [\![\, judgment(j) \in \{\text{R,X}\} \,]\!]}{500}$$

As can be expected, confidence-weighted scores increase as judgments become less strict. In particular, allowing for unsupported answers has a strong impact on the scoring. Comparing run UAmsT11qaM1 (using the top 50 documents) with UAmsT11qaM2 (using the top 100 documents), indicates that using a smaller set of documents for answer selection is to be preferred; although this conclusion is not supported by CWS(R,U,X).

Runs UAmsT11qaM2 and UAmsT11qaM3 both use the top 100 documents, but we did not sort the responses in UAmsT11qaM3 with respect to confidence. This was meant to evaluate our confidence score computation algorithm. The results in Table 4 are very inconclusive, as UAmsT11qaM2 scores better for CWS(R,U) and CWS(R,U,X) but worse for CWS(R) and CWS(R,X).

In addition, we also calculated the precision of each run, neglecting confidence weights. E.g.,

$$\text{Prec(R)} = \frac{\sum_{i=1}^{500} [\![\, judgment(j) = \text{R} \,]\!]}{500}$$

The average precision scores are displayed in Table 5.

Table 5: Summary of the avg. precision for the main task.

UAmsT10qa...	M1	M2	M3
Prec(R)	0.128	0.112	0.112
Prec(R,U)	0.170	0.176	0.176
Prec(R,X)	0.134	0.132	0.132
Prec(R,U,X)	0.176	0.196	0.196

As with the confidence-weighted scores, precision also increases as judging becomes less strict. Again, counting unsupported answers as correct has the strongest impact on precision. Note, that UAmsT11qaM2 and UAmsT11qaM3 have the same scores for all judgments since they differ only with respect to confidence sorting. The higher precision scores of UAmsT11qaM2 and UAmsT11qaM3 compared to UAmsT11qaM1, when allowing for unsupported answers, are probably due to the lower number of NIL answers: UAmsT11qaM1 contains 234 questions having NIL as an answer, whereas UAmsT11qaM2 and UAmsT11qaM3 contain only 88 questions having NIL as an answer.

Table 6: Analysis of the scores for **UAmsT11qaM1**.

Question class	% quest.	Prec.	CWS	CWS diff.
agent	5.8%	0.172	0.150	+3.4%
aka	3.0%	0.133	0.131	−9.6%
capital	0.6%	0	0.136	−6.2%
date	16.2%	0.160	0.160	+10.3%
date-birth	2.0%	0.300	0.164	+13.1%
date-death	1.2%	0.833	0.165	+13.7%
expand-abbr	1.8%	0	0.132	−8.9%
location	14.4%	0.097	0.148	+2.0%
name	4.8%	0.041	0.133	−8.2%
number-dist	1.4%	0	0.163	+12.4%
number-height	2.0%	0.200	0.131	−9.6%
number-length	0.4%	0	0.145	±0%
number-many	1.0%	0.200	0.158	+8.9%
number-people	0.8%	0	0.135	−6.8%
number-money	0.8%	0.250	0.175	+20.6%
number-much	1.8%	0.111	0.132	−8.9%
number-speed	0.6%	0.666	0.155	+6.9%
number-age	1.2%	0.166	0.155	+6.9%
object	1.4%	0.428	0.132	−8.9%
pers-def	0.8%	0.250	0.132	−8.9%
pers-ident	4.4%	0.090	0.141	−2.7%
thing-def	0.2%	0	0.136	−6.2%
thing-ident	16.2%	0.061	0.133	−8.2%
what-np	14.8%	0.121	0.143	−1.3%
unknown	2.4%	0	0.133	−8.2%
Total		0.128	0.145	

Table 6 offers a closer look at our primary run for the main task, UAmsT11qaM1, and provides a breakdown in terms of the individual question types. Column 1 lists the question classes as discussed in Section 3.1.3 which have at least one question in the TREC 2002 question set; column 2 lists the percentage of questions belonging to a particular class. In column 3 the individual precision scores are displayed. Column 4 lists the confidence-weighted scores for each class of questions. The last column records the relative difference between the mean CWS for the class and the overall CWS for the run (shown at the bottom of column 4). All confidence-weighted scores are based on strict evaluation, i.e., CWS(R).

4 Web Track

TREC 2002's Web track features two tasks, named page finding and topic distillation, using a recent crawl of the .gov domain (January 2002). For the named-page finding task, we experimented with plain text runs, anchor-text runs, and their combinations. For topic distillation task, we additionally experimented with ways to exploit the link and URL structure in the collection.

The remainder of this section is organized as follows. After discussing some key facts about the collection and our experimental set-up, we describe our runs for the named pages finding task, and for the topic distillation task, and then discuss our findings on the link structure of the collection.

4.1 The .GOV Collection

The size of the .GOV collection, 1.25 million documents and in total 18 gigabytes, posed a challenge for our FlexIR system. Although CSIRO did a commendable job in preparing this collection, we occasionally stumbled upon binary content, and extremely long strings of characters. We had to implement various modifications to overcome the linux filesize limits. The resulting text-based index is 6 Gb (3.25 Gb for the index and 2.5 Gb for the inverted index).

We built two separate indexes for the .GOV collection: a text-only index, and an anchor-text index. For the free-text index, we indexed all of the documents' textual contents, decoding special html-characters into plain ASCII, and replacing diacritics with the unmarked characters. We used the Porter stemmer [17], and a stoplist of 391 words. Our text index contains 1,247,753 documents. We also built a separate anchor-text only index, assigning the anchor-texts to the linked documents. Again, we used the Porter stemmer. Our anchor-text index contains 667,737 documents, which is 53.51% of the text-based index. For the retrieval runs, we experimented with two weighting schemes, the familiar Lnu.ltc scheme and a scheme, baptized Lnm.ltc, based on minimal matching span (MSM) weighting (see section 3.1.2 for details). We did not use blind feedback in any of our runs.

4.2 Named Page Finding Task

For the named page finding task, there are 150 short queries containing the name of a page. The average query length is 3.81 words or 3.55 words after removing stopwords. There is considerable ambiguity when retrieving a unique page characterized by such a short query. As it turned out, there is a unique relevant page for 132 of the topics, for 16 topics there are two relevant pages, and there are three relevant pages for the remaining 2 topics.

The precursor of this task was TREC 2001's home page finding task [9]. For entry page finding, non-content features such as URLs and links provided valuable information [11]. We did not see a straightforward way to use non-content features for this year's task. An alternative is to use the anchor-texts in the collection [5]. For the named page finding task, we experimented with plain text runs, anchor-text runs, and their combinations.

Table 7: Overview of the named page finding runs.

Run	Type	Weighting
1. UAmsT02WnTl	Text-only	Lnu.ltc
2. UAmsT02WnTm	Text-only	Lnm.ltc
3. UAmsT02WnA	Anchors-only	Lnu.ltc
4. UAmsT02WnTlA	Combined 1/3	
5. UAmsT02WnTmA	Combined 2/3	

The submitted runs are shown in Table 7. The text and anchor-only runs were combined in the following manner. We only considered the first ten results of both runs; following Lee [13], the scores are normalized using $RSV_i' = \frac{RSV_i - min_i}{max_i - min_i}$. We assigned new weights to the documents using the summation function used by Fox and Shaw [7]: $RSV_{new} = RSV_1 + RSV_2$.

Table 8: Anchor-text only runs.

Run	MRR	Top 10	Unknown
UAmsT02WnTl	0.4254	82 (54.7%)	46 (30.7%)
UAmsT02WnTm	0.2601	58 (38.7%)	83 (55.3%)
UAmsT02WnA	0.3279	69 (46.0%)	70 (46.7%)
UAmsT02WnTlA	**0.4317**	99 (66.0%)	35 (23.3%)
UAmsT02WnTmA	0.3672	81 (54.0%)	59 (39.3%)

The results for our official run are shown in Table 8; the column labeled 'MRR' lists the mean reciprocal rank of the first correct answer (the official measure); the column labeled 'Top 10' lists the number of topics with a correct named pages in the top 10; and the column labeled 'Unknown' lists the number of topics for which no named page was found in the top 50.

The results show that the text runs using Lnu.ltc weighting scheme were more effective than those using the Lnm.ltc scheme. The combined text and anchor-text run performed the best with an MRR of 0.4317. The anchor-text only run, which indexes only half of the documents, scores 77.08% of the text only run. The combination of both runs improves the MRR by 1.48% over the text only run; the number of topics in the top 10 is improved by 20.73% over the text only run.

4.3 Topic Distillation Task

For topic distillation, only key resources in the collection will be regarded as relevant. A page can be a key resource solely by its set of links, e.g., a home page of a relevant site. The challenge is to find ways to exploit the additional structure in the documents. There are 50 topics, having on average 3.24 words (2.92 after removing stop words). Although key resources are supposedly much rarer than relevant documents, there turn out to be on average 32.12 key resources per topic.[1]

Similar to the named page finding task, we created runs using the text-only and anchors-only collections (see Table 9 for an overview of the official runs). We experimented with the following approach for exploiting the URL information (indicated as 'base URL clusters' in Table 9). Since there will rarely be more than one key resource per site, we cluster pages by their base URL, and return the page with the lowest URL depth. Specifically, we assign the top 100 documents

Table 9: Overview of the topic distillation runs.

Run	Type	Weighting
1. UAmsT02WtT	Text	Lnm.ltc
2. UAmsT02WtTri	Realized indegree 1	
3. UAmsT02WtA	Anchors	Lnu.ltc
4. UAmsT02WtAri	Realized indegree 3	
5. UAmsT02WtAcs	Base URL clusters 3	

[1] This is over 49 topics, ignoring Topic 582 for which there were no key resourses in the collection. There are 11 topics with less than 10 key resources.

to the first 10 different base URLs. Next, we return the page with the lowest URL depth or slash-count per cluster.

We also experimented with the use of the link structure of the documents (indicated as 'realized indegree' in Table 9). There exist approaches that look at the global link structure, i.e., page-rank [2], and those that look at the local link structure surrounding an initially retrieved set of documents, i.e., Hyperlink Induced Topic Search (HITS) [10]. We follow Kleinberg [10] in considering the local set of pages containing the initially retrieved documents, plus all documents linked from, or linking to documents in this set. For the anchor text runs we used the top 100 results, and for the text runs, the local set is determined by the top 200 documents. We implemented an approach that combines both global and local link structure by comparing how much of the links of a page are present in the local set of initially retrieved documents. Specifically, we calculate the local indegree (the number of a page's incoming links that are in the local set) divided by the page's indegree (the total number of links to a page). This number, which gives an indication of the topicality, is multiplied by the local indegree. The (local) indegree by itself gives an indication of the relative importance of the page [1]. The resulting new ranking is solely based on the structural link information.

Table 10: Official topic distillation run results.

Run	Prec. at 10, 20, and 30		
UAmsT02WtT	**0.1755**	0.1245	0.1020
UAmsT02WtTri	0.0673	0.0582	0.0463
UAmsT02WtA	0.1000	0.0714	0.0558
UAmsT02WtAri	0.0633	0.0469	0.0381
UAmsT02WtAcs	0.0653	0.0786	0.0660

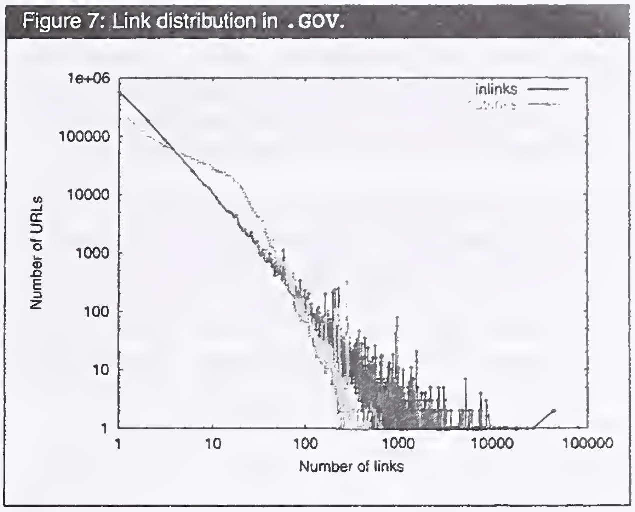
The results of our official runs are shown in Table 10. The official measure is precision at 10, at which the text-only run scores best with 0.1755. The anchor-text only run, covering only half the documents, scores 56.98% of the text only run. A text only run using Lnu.ltc weighting, not submitted, scored better than the official run, with a precision at 10 of 0.2102. The run using the base URL clusters fails to improve the anchor-text base run, although it improves precision at 20 and 30. The runs based on link information all perform worse than the underlying base runs.

4.4 Link Structure

The link structure in .GOV should be a fairly representative sample of the current Internet.[2] Figure 7 shows the link distribution in the .GOV collection on a logarithmic scale. Both the distribution of outlinks, and the distribution of inlinks show a powerlaw behavior as observed by [6]. The five pages with the highest number of outlinks are:

- visibleearth.nasa.gov/browse.html (653);

Figure 7: Link distribution in .GOV.



- www.bls.gov/oes/2000/oes_alph.htm (647);
- www.bls.gov/oes/2000/oes_stru.htm (646);
- hn.usatlas.bnl.gov/cgi-bin/cvsweb.cgi/offline/graphics/Jive/ (548);
- www.whitehouse.gov/news/nominations/index-date.html (471);

The five pages with the highest number of inlinks are:

- www.usgs.gov/ (44,499);
- www.usda.gov/ (43,324);
- www.nasa.gov/ (26,693);
- www.usda.gov/news/privacy.htm (23,418);
- www.usgs.gov/accessibility.html (23,234);

It is of crucial importance for link-based approaches to be able to distinguish between intrinsic links (links within a site, mainly for navigational purposes) and transverse links (links between sites). The .GOV contains in total 11,164,829 links between pages in the collection. We first identified the site of a page as it base URL, with the removal of any prefix starting with www. This results in a set of 2,413,054 transverse links (or 22%). This reduced set still contained many within-site links, so we further reduced the set by removing links between base URLs when either is a substring of the other. For example, a link between www.nih.gov and www.nlm.nih.gov regarded as instrinsic, while a link between www.nlm.nih.gov and www.nichd.nih.gov is regarded as transverse. The resulting set of transverse links contains 1,699,834 links (or 15% of all links).

Arguably, pages that do not receive links from other sites will rarely be key resourses. This motivated experiments with anchor-text only runs on three different indexes:

First Anchors Index Only extracting complete link descriptions in the collection. This includes all transverse links, and only a small proportion of intrinsic links (which are usually included as relative locations). All unique anchor-texts are assigned to the document to which the

[2] We used the provided links_id and id2url files. These contain a few bugs, e.g., G15-52-0622377 is listed as www.lib.noaa.govnewj.htm instead of www.lib.noaa.gov/edocs/..newj.htm.

link points. Considering the dramatic difference in the number of inlinks discussed above, we decided to remove repeated occurrences of the same anchor-text. This resulted in a set of 313,562 anchor-texts covering 186,328 documents, only 15% of the collection.

Second Anchors Index Here we try to recover as many links as possible, by unfolding relative links based on the URL path of the page in which the link occurs, and simplying the resulting URL paths. This includes both intrinsic and transverse links. We again remove repeated occurrences of the same anchor-texts. The result is a set of 1,110,566 anchor-texts covering 667,737 documents, which is 54% of the collection.

Third Anchors Index We use the same procedure as for the second anchors index, but now retain all links as they appear in the collection. Thus, if the same anchor-text occurs thousands of times, we include it thousands of times (similar to [5]). The resulting index is based on 2,766,946 anchor-texts covering 667,737 documents, which is 54% of the collection.

Table 11: Anchors only run results.

Run	Index	MRR	Prec. at 10
UAmsT02WnA'	Anchors 1.	0.1391	
UAmsT02WnA	Anchors 2.	**0.3279**	
UAmsT02WnA"	Anchors 3.	0.3098	
UAmsT02WtA'	Anchors 1.		0.0673
UAmsT02WtA	Anchors 2.		**0.1000**
UAmsT02WtA"	Anchors 3.		0.0837

The post-submission experiments shown in Table 11 show the performance of anchor-text only runs using the three anchor-text indexes. The second anchor-text index, which was used for our official runs, shows the best performance.

We carried out pre-submission experiments using Kleinberg's HITS [10] in order to retrieve key resources for the topic distillation task. Table 12 shows the results for the test topic '*obesity in the U.S.*': the 'Base top 10' are the top 10 results of the text base run; and 'HITS 100' and 'HITS 200' show the top 10 authorities over the top 100 and top 200 documents respectively. Although HITS is successful at isolating key resources, there is a considerable topic drift towards generally good 'authorities.' As is well-known, good authorities and the number of inlinks show considerable correlation [10, 1]. Thus, one can easily image how a loosely-related site with a high indegree can infiltrate in the HITS method. We experimented with a link-based method that tries to avoid such topic drift, by looking at the proportion of inlinks that is in the local set of documents. The top 10 results are also shown in Table 12: 'Realized indegree 100' and 'Realized indegree 200' show the results over the top 100 and top 200 documents of the initial text base run. Informal evaluation shows that our combined approach is much more robust than HITS (by comparing results over different numbers of top documents), for example, when considering the top 500 initially retrieved documents HITS authorities appear almost unrelated to the topics, where as the 'realized indegree' method is still on topic.

Table 12: Test Topic "obesity in the U.S."

Base Top 10
www.surgeongeneral.gov/topics/obesity/calltoaction/4_2.htm
4woman.gov/faq/easyread/obesity-etr.htm
whi.nih.gov/guidelines/obesity/e_txtbk/intro/intro.htm
www.surgeongeneral.gov/topics/obesity/calltoaction/2_0.htm
www.surgeongeneral.gov/topics/obesity/calltoaction/fact_glance.htm
www.surgeongeneral.gov/topics/obesity/calltoaction/principles.htm
www.cdc.gov/nccdphp/dnpa/obesity/trend/maps/
www.nalusda.gov/ttic/tektran/data/000010/76/0000107699.html
www.nalusda.gov/ttic/tektran/data/000010/09/0000100959.html
www.surgeongeneral.gov/topics/obesity/calltoaction/2_2.htm
HITS Top 100
www.nih.gov/icd/od/foia/
www.nlm.nih.gov/
www.nlm.nih.gov/medlineplus/obesity.html
www.nlm.nih.gov/accessibility.html
www.nlm.nih.gov/contacts/
www.nlm.nih.gov/disclaimer.html
www.nichd.nih.gov/
www.nlm.nih.gov/medlineplus/diabetes.html
www.nlm.nih.gov/medlineplus/highbloodpressure.html
www.nlm.nih.gov/medlineplus/sleepdisorders.html
HITS Top 200
www.nih.gov/icd/od/foia/
www.nlm.nih.gov/
www.nlm.nih.gov/medlineplus/obesity.html
www.nichd.nih.gov/
www.nlm.nih.gov/disclaimer.html
www.nlm.nih.gov/accessibility.html
www.nlm.nih.gov/contacts/
www.nlm.nih.gov/medlineplus/diabetes.html
www.nlm.nih.gov/medlineplus/highbloodpressure.html
www.nlm.nih.gov/medlineplus/respiratorydiseasesgeneral.html
Realized Indegree Top 100
www.niddk.nih.gov/health/nutrit/pubs/unders.htm
www.nlm.nih.gov/medlineplus/obesity.html
hin.nhlbi.nih.gov/bmi_palm.htm
www.ahcpr.gov/research/may00/0500RA6.htm
www.nhlbi.nih.gov/guidelines/obesity/bmi_tbl.htm
www.nlm.nih.gov/medlineplus/diabetes.html
www.fitness.gov/Reading_Room/reading_room.html
www.cdc.gov/nccdphp/dnpa/dnpalink.htm
response.restoration.noaa.gov/photos/dispers/dispers.html
www.fda.gov/bbs/topics/NEWS/NEW00575.html
Realized Indegree Top 200
www.nhlbi.nih.gov/health/public/heart/obesity/lose_wt/patmats.htm
www.nlm.nih.gov/medlineplus/obesity.html
hin.nhlbi.nih.gov/bmi_palm.htm
www.niddk.nih.gov/health/nutrit/pubs/unders.htm
www.ftc.gov/bcp/conline/pubs/health/setgoals.htm
www.cdc.gov/nccdphp/dnpa/
www.cdc.gov/health/obesity.htm
whi.nih.gov/health/prof/heart/
www.ahcpr.gov/research/may00/0500RA6.htm
www.ftc.gov/bcp/conline/pubs/health/setgoals.pdf

Earlier attempts at exploiting link structure (in the ad hoc task) failed to show an improvement of retrieval effectiveness [9]. Our experiments with HITS and with the 'realized indegree' method show a decrease in precision at 10 (see Table 10). A possible explanation could be the topics used for the distillation task. These are more specific than the very general topics used in [10], such as '*java*,' '*censorship*,' '*search engines*,' and '*Gates*.' Also, after stopping, the test topic '*obesity in the U.S.*' results in the one-word query '*obesity*.' For such general queries, relevant documents will dominate the top 10, top 100, or even top 200 of initially retrieved documents. Under this assumption, link-based approaches, which ignore the content of documents and solely consider the link topology, can be effective. If non-relevant documents dominate the initially retrieved set of documents, one cannot expect link-based methods to deliver.

5 Conclusions

In this paper we described our participation in the TREC 2002 Novelty, Question answering, and Web tracks. We set up a baseline system for the Novelty track, and showed that both lemmatizing and combining yield significant improvements for the relevance as well as the novely part. We can look at the novelty part of our system in isolation by assuming perfect output from the relevance part of our system. As it turns out, our system's recall scores for the novelty part are very close to the maximal performance. Our results for the relevance part of the task are less impressive. It seems that the relevance part is the hardest and most important part of the task.

For the question answering track, we experimented with a revised version of our Tequesta system. The main innovation was to introduce document retrieval techniques that were tuned for question answering purposes; in particular, we used high precision settings, together with minimal span matching for each document. In a later stage of the question answering process, MSMs are used to restrict documents to passages which are likely to contain the answer. Our results show considerable differences across question types, which is probably due to quality of the extraction components.

For the web track, we set up a baseline system using separate text and anchor-text indexes. We experimented with the use of non-content features, such as the URL and link structure in the collection for the topic distillation task. Our results failed to show a positive effect on retrieval effectiveness. For the named page finding task, a genuine needle-in-a-haystack task, we experimented with text-only and anchor-text only runs, and their combinations. Here, the combined text/anchor-text run slightly improves the mean reciprocal rank, but significantlty improves the number of topics with the named page in the top 10.

Acknowledgments

Jaap Kamps was supported by the Netherlands Organization for Scientific Research (NWO), grant # 400-20-036. Christof Monz was supported by the Physical Sciences Council with financial support from NWO, project 612-13-001. Maarten de Rijke was supported by grants from NWO, under project numbers 612-13-001, 365-20-005, 612.069.006, 612.000.106, 220-80-001, and 612.000.207.

References

[1] B. Amento, L. Terveen, and W. Hill. Does 'authority' mean quality? predicting expert quality ratings of web documents. In E. Yannakoudakis, N.J. Belkin, M.-K. Leong, and P. Ingwersen, editors, *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 296–303, 2000.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World Wide Web Conference*, pages 107–117, 1998.

[3] C. Buckley, A. Singhal, and M. Mitra. New retrieval approaches using SMART: TREC 4. In D. Harman, editor, *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 25–48. NIST Special Publication 500-236, 1995.

[4] C. Clarke, G. Cormack, and T. Lynam. Exploiting redundancy in question answering. In Kraft et al. [12], pages 358–365.

[5] N. Craswell, D. Hawking, and S. Robertson. Effective site finding using link anchor information. In Kraft et al. [12], pages 250–257.

[6] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM*, pages 251–262, 1999.

[7] E.A. Fox and J.A. Shaw. Combination of multiple searches. In D.K. Harman, editor, *The Second Text Retrieval Conference (TREC-2)*, pages 243–252. National Institute for Standards and Technology. NIST Special Publication 500-215, 1994.

[8] D.K. Harman. Overview of the TREC 2002 Novelty Track. In *This Volume*.

[9] D. Hawking and N. Craswell. Overview of the TREC-2001 web track. In Voorhees and Harman [21], pages 25–31.

[10] J.M. Kleinberg. Authoritative structures in a hyperlinked environment. *Journal of the ACM*, 46:604–632, 1999.

[11] W. Kraaij, T. Westerveld, and D. Hiemstra. The importance of prior probabilities for entry page search. In K. Järvelin, M. Beaulieu, R. Baeza-Yates, and S.H. Myaeng, editors, *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and development in information retrieval*, pages 27–34, 2002.

[12] D.H. Kraft, W.B. Croft, D.J. Harper, and J. Zobel, editors. *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[13] J.H. Lee. Combining multiple evidence from different properties of weighting schemes. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 180–188, 1995.

[14] D. Lin. PRINCIPAR—an efficient, broad-coverage, principle-based parser. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*, pages 42–48, 1994.

[15] C. Monz and M. de Rijke. Shallow morphological analysis in monolingual information retrieval for Dutch, German and Italian. In C. Peters, M. Braschler, J. Gonzalo, and M. Kluck, editors, *Proceedings CLEF 2001*, LNCS 2406, pages 262–277. Springer Verlag, 2002.

[16] C. Monz and M. de Rijke. Tequesta: The University of Amsterdam's texual question answering system. In Voorhees and Harman [21], pages 519–528.

[17] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[18] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[19] H. Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, 1994.

[20] The TREC 2002 novelty track guidelines. `http://trec.nist.gov/act_part/guidelines/novelty_guidelines.html`.

[21] E.M. Voorhees and D.K. Harman, editors. *The Tenth Text Retrieval Conference (TREC 2001)*. National Institute for Standards and Technology. NIST Special Publication 500-250, 2002.

Automatic Shot Boundary Detection and Classification of Indoor and Outdoor Scenes

A. Miene, Th. Hermes, G. Ioannidis, R. Fathi, and O. Herzog

TZI - Center for Computing Technologies
University of Bremen
Universitätsallee 21-23
D-28359 Bremen, Germany
{andrea|hermes|gtis|fathi|herzog}@tzi.de

Abstract. This paper describes our contribution to the TREC 2002 video analysis track. We participated in the shot detection task and in the feature extraction task (features indoors and outdoors).
The shot detection approach is based on histogram differences and uses adaptive thresholds. Multiple detected shot boundaries that follow each other within a short temporal interval are grouped together and classified as a gradual change beginning with the first and ending with the last shot boundary in the interval.
For the feature extraction task we examined whether it is possible to classify indoor and outdoor shots by their color distribution. In order to analyze the color distribution we use first order statistical features. The shots are classified into indoor and outdoor shots using a neural net.

1 Introduction

The Center for Computing Technologies (TZI), University of Bremen, Germany, participated in the video analysis track in the shot detection task and in the feature extraction task (features indoors and outdoors).

The shot detection approach is based on histogram differences. It is divided into two steps - feature extraction and shot boundary detection. Firstly, the histogram differences are calculated for the entire video in real time. Secondly, shot boundaries are detected. The advantage of this approach is the possibility to set adaptive thresholds for the shot boundary detection considering all extracted features of the complete video sequence. The adaptive threshold is set to a percentage of the maximum of all calculated difference values of the video. In the case of gradual changes, often multiple shot boundaries are detected. Therefore multiple detected shot boundaries that follow each other within a short temporal interval are grouped together and a gradual change is detected beginning with the first and ending with the last shot boundary in the interval. The approach is explained in more detail in section 2.

To extract the features indoors and outdoors we use a feed forward neural net as a classificator, trained by a backpropagation learning rule. The input is a feature vector describing the color distribution of an image. The output is the probability for each feature (indoors and outdoors) to appear in the image. The approach is discussed in section 3.

2 Shot detection

Quite a lot of approaches to shot boundary detection were proposed in the literature. An overview is given in [Lienhart, 1999,Yusoff et al., 1998]. The principle methodology of shot boundary detection is to extract one or more features from every nth frame of a video sequence, to compute the difference of features for consecutive frames, and to compare these differences to a given threshold. Each time the threshold is exceeded a shot boundary is detected. The various approaches differ concerning the used features.

The shot boundary detection system we used for TREC 2002 is based on the approach presented in [Miene et al., 2001]. As mentioned before , the approach can be divided into two main parts. The first part is to extract all needed features from a video. The second part is to detect the shot boundaries based on the previously extracted features.

For the feature extraction part each frame is converted into a grayscale image. Then a histogram H_G is created. Subsequently, the squared differences between each two consecutive frames

$$H_{G_{Diff}}(n, n-1) = \sum_{i=0}^{255} \frac{(H_G(n)(i) - H_G(n-1)(i))^2}{Max(H_G(n)(i), H_G(n-1)(i))} \tag{1}$$

are calculated. $H_G(n)(i)$ denotes a grayscale histogram value at index i of frame n. $Max(H_G(n)(i), H_G(n-1)(i))$ denotes the maximum of both grayscale histogram values $H_{Gray}(n)(i)$ and $H_{Gray}(n-1)(i)$, and is used as a normalization factor.

This leads to a feature difference list in order to detect shot boundaries, which is compared to a threshold. To determine the adaptive threshold, the maximum of all calculated difference values of the actual video is calculated. The adaptive threshold for the actual video is specified as a percental value of the maximum:

$$Th = \frac{Max\{H_{G_{Diff}}(1,0), \ldots, H_{G_{Diff}}(n, n-1)\} \cdot Th_{percentage}}{100} \tag{2}$$

For gradual changes like dissolves or wipes the shot boundary detection often detects more than one boundary per shot. Therefore, all shot boundaries which belong to the same shot have to be merged into one boundary. This step is illustrated in Figure 1. Shot boundaries are merged together if the temporal distance between their occurrences is less than a threshold. The minimal frame number of the merged shot boundaries determines the start, and the maximum frame number determines the end of the gradual change. The exact boundary position is set to the maximum feature difference value within the merged shot boundaries.

Before preparing our results for TREC we tested our shot detection on three videos from the feature developement collection, for which we determined the

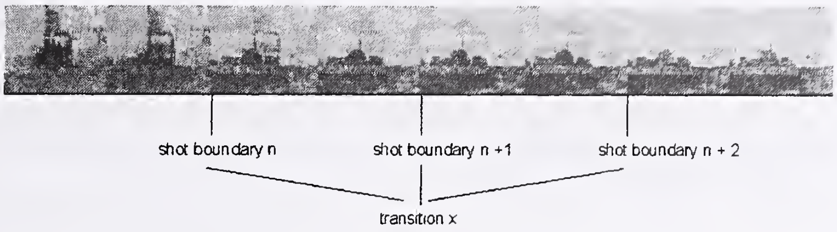


Fig. 1. Merging of multiple detected shot boundaries[Miene et al., 2001].

shot boundaries manually. The results of this experiment are shown in table 2. *File* denotes the file name of the video, *human* the amount shot boundaries determined manually, *auto* the total number of shot boundaries detected by the system, *correct* the amount of correct detected shot boundaries, *false* the amount of false alarms and *missing* the amount of shot boundaries, that were not detected by the system. The last two colums contain the percentage values for precission and recall. For this test we did not distinguish between hard cuts and gradual changes.

file	human	auto	correct	false	missing	recall	precission
00028a.mpg	116	79	79	0	37	68.1	100
00435.mpg	108	54	47	7	61	42.6	85.2
00535.mpg	120	108	100	8	20	83.3	92.6
over all	344	241	226	15	118	65.7	93.8

In the following section we present our approach for the classification of indoor and outdoor scenes.

3 Classification of indoor and outdoor scenes

For the feature extraction task we have examined whether it is possible to classify indoor and outdoor shots by their color distribution. In order to analyze the color distribution, first order statistical features are used, which are extracted from the histograms of the three color channels (RGB) and the grey level histogram. The features calculated from each histogram are average, variance, and amount of peaks, normalized to an interval $[0.0, \ldots, 1.0]$. Therefore we calculate 12 statistical features alltogether. In order to classify the shots into indoor and outdoor shots, a feed forward neural net with backpropagation learning was trained. For this task we used the SNNS (Stuttgart Neural Network Simulator) [SNNSv4.2, 2002].

At the input layer the 12 statistical features are presented, that were obtained from the histograms. The output layer consists of two neurons that take on values between 0.0 and 1.0 measuring the probability for the features indoors or outdoors in the shot. Two hidden layers with 20 neurons each are initialized with random weights. In order to train the neural net, some videos from the feature development collection were chosen. The shots are classified manually to generate 323 training data sets, 178 for indoors, and 145 for outdoors. Figure 2 shows the trained neural net.

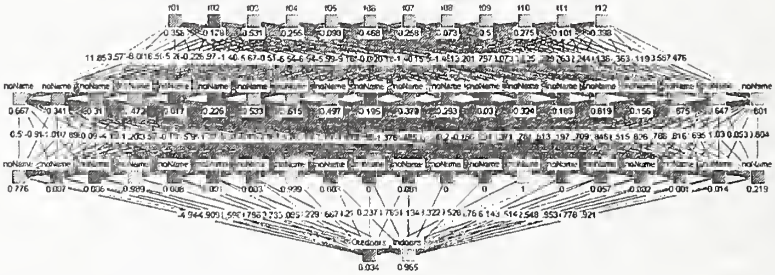


Fig. 2. Trained neural net.

In order to classify the shots from the feature extraction test collection, a set of n key frames is extracted from each shot. Every kth frame of a shot is used as a key frame, but in order to be more independent of inaccuracies during the shot detection and of gradual changes (e.g., wipes, fades, or dissolves) a number of frames around the shot boundaries is skipped (see Figure 3). In order to classify a shot, the set of n key frames is presented to the neural net.

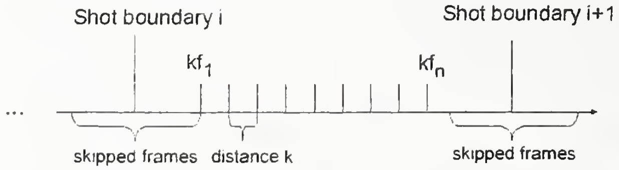


Fig. 3. Extraction of key frames.

For each of the two output neurons a list is obtained containing n values, one for each key frame. The median is calculated for each list to obtain the final indoors or outdoors probabilities for the shot. In order to measure the accuracy of the classification result, the difference between the median values of the indoors and the outdoors neuron is calculated. If the difference exceeds a threshold the shot is classified to contain the feature with the higher probability. The difference is also used for the ranking.

4 Future Work

For the next months and also for our next participation in TREC 2003 we will concentrate on the improvement of our shot detection approach concerning the detection of gradual changes. We have also to examine how to obtain better results for the extraction of the indoors and outdoors features. As mentioned before the neural net was trained with indoor and outdoor example frames from the feature development collection. The results from the TREC evaluation of our results shows that there are serious problems with the classification of the material from the feature test collection. At the moment we are working on the analysis of these problems. One major problem appears to be, that we trained the net only with examples of indoor and outdoor scenes. Therefore the results for scenes containing neither indoor nor outdoor scenes are undefined. Therefore, especially artificial scenes lead to a wrong classification.

In addition we are looking forward to develop further modules for our feature extraction system to be able to extract other features like text or human faces.

References

[Lienhart, 1999] Lienhart, R. (1999). Comparison of Automatic Shot Boundary Detection Algorithms. In *Proc. SPIE Vol. 3656 Storage and Retrieval for Image and Video Databases VII*, pages 290–301, San Jose, CA, USA.

[Miene et al., 2001] Miene, A., Dammeyer, A., Hermes, T., and Herzog, O. (2001). Advanced and adapted shot boundary detection. In Fellner, D. W.. Fuhr, N., and Witten, I., editors, *Proc. of ECDL WS Generalized Documents*. pages 39–43.

[SNNSv4.2, 2002] SNNSv4.2 (2002). *SNNS Stuttgart Neuronal Network Simulator User Manual, Version 4.2.* University of Stuttgart and University of Tübingen. http://www-ra.informatik.uni-tuebingen.de/downloads/SNNS/SNNSv4.2.Manual.pdf.

[Yusoff et al., 1998] Yusoff, Y., Christmas, W., and Kittler, J. (1998). A study on automatic shot change detection. *Lecture Notes in Computer Science*, 1425.

Information by Computer. Addison Wesley Publishing, New York.

4. TREC-9, 2000. Call for participation Text Retrieval Conference 2000 (TREC-9).
5. TREC-10, 2001. Call for participation Text Retrieval Conference 2001 (TREC-10).
6. Vicedo J.L. *SEMQA: Un modelo semántico aplicado a los sistemas de Búsqueda de Respuestas*. Phd Thesis. May 2002.
7. Vicedo J.L., Ferrandez A. And Llopis F. *University of Alicante at TREC-10*. In Proceedings of the Tenth Text Retrieval Conference. November 2001. Gaithersburg (USA).
8. Vicedo J.L. and Ferrandez A. *A semantic approach to Question Answering systems*. In Proceedings of the Ninth Text Retrieval Conference. November 2000. Gaithersburg (USA).
9. Vicedo J.L. *Using semantics for Paragraph selection in Question Answering systems*. In proceedings of the Proceedings of the Eighth String Processing and Information Retrieval Conference (SPIRE'2001). November 2001. Laguna de San Rafael (Chile).

Building a Foundation System for Producing Short Answers to Factual Questions

Sameer S. Pradhan*, **Gabriel Illouz**†§, **Sasha J. Blair-Goldensohn**†, **Andrew Hazen Schlaikjer**†, **Valerie Krugler***, **Elena Filatova**†, **Pablo A. Duboue**†, **Hong Yu**†, **Rebecca J. Passonneau**†, **Steven Bethard***, **Vasileios Hatzivassiloglou**†, **Wayne Ward***, **Dan Jurafsky***, **Kathleen R. McKeown**†, and **James H. Martin***

*Center for Spoken Language Research
University of Colorado
Boulder, CO 80309, USA

†Department of Computer Science
Columbia University
New York, NY 10027, USA

Abstract

In this paper we describe question answering research being pursued as a joint project between Columbia University and the University of Colorado at Boulder as part of ARDA's AQUAINT program. As a foundation for targeting complex questions involving opinions, events, and paragraph-length answers, we recently built two systems for answering short factual questions. We submitted results from the two systems to TREC's Q&A track, and the bulk of this paper describes the methods used in building each system and the results obtained. We conclude by discussing current work aiming at combining modules from the two systems in a unified, more accurate system and adding capabilities for producing complex answers in addition to short ones.

1 Introduction

The Department of Computer Science at Columbia University (CUCS) and the Center for Spoken Language Research (CSLR) at the University of Colorado at Boulder are collaborating to develop new technologies for question answering. This project is supported by the ARDA AQUAINT (Advanced QUestion Answering for INTelligence) program. The project plans to integrate robust semantics, event detection, information fusion, and summarization technologies to enable a multimedia question answering system. The goal is to develop a system capable of answering complex questions; these are questions that require interacting with the user to refine and clarify the context of the question, whose answer may be located in non-homogeneous databases of speech and text, and for which presenting the answer requires combining and summarizing information from multiple sources and over time. Generating a satisfactory answer to complex questions requires the ability to collect all relevant answers from multiple documents in different media, weigh their relative importance, and generate a coherent summary of the multiple facts and opinions reported. In order to achieve these goals, we are developing and integrating four core technologies: semantic annotation (CSLR), context management (CSLR), event recognition and information tracking (Columbia), and information fusion and summary generation (Columbia).

Prior to the start of this project, neither site had developed a complete question answering system, so we had to build the foundation components and put them together in approximately six months. We elected to build two independent systems as a first step, one at each site, rather than attempting to integrate components across the two sites in

§Currently at LIMSI-CNRS, Orsay, France.

such a short period. This gave us the added benefit that each site was able to focus more of their effort on specific components they were interested in, and avoided the need for developing complex protocols for communication between the modules across the sites.

The paper focuses on our development of our foundation question answering systems for TREC, processing factual questions only and producing short answers. Most of the remainder of the paper (Sections 2 and 3) discusses the two architectures we developed, ways that each departs from standard question answering methodology, and our results on this year's TREC questions. We submitted the results of these two architectures as runs 1 and 2 with the tag `cuaq`. We conclude with a discussion of our current integration effort and ongoing work on adding advanced components for handling additional question types to our foundation system.

2 System A — The Boulder System

The novel feature of our approach in System A is the use of shallow semantic representations to enhance potential answer identification. Most successful systems first identify a list of potential answer candidates using pure word-based metrics. Syntactic and semantic information at varying granularity is then used to re-rank those candidates [10, 11]. However, most of these semantic units are quite specific in what they label. We identify a small set of *thematic roles*—viz., *agent*, *patient*, *manner*, *degree*, *cause*, *result*, *location*, *temporal*, *force*. *goal*, *path*, *percept*, *proposition*, *source*, *state*, and *topic*—in the candidate answer sentences, using a statistical classifier [9]. The classifier is trained on the FrameNet database [2].

2.1 Architecture

The following sequence of actions will be taken in response to an input query:

1. Classify the question according to type by identifying the Named Entity and Thematic Role of the expected answer type. This also defines a set of answer type patterns, and includes named entity tagging and parsing the question for thematic roles.
2. Identify the focus, i.e., certain salient words or phrases in the question that are very likely to be present in the answer string in one form or the other.
3. Extract a set of query words from the question, and apply semantic expansion to them.
4. Submit the query words to the mg (Managing Gigabytes) [15] IR engine and get back a rank-ordered set of documents.
5. Keep the top M (approximately 500) documents and prune the rest.
6. Segment documents into paragraphs and prune all but the top N paragraphs (approximately 2.500).
7. Generate scoring features for the paragraphs, including named entity tagging and parsing of paragraphs to add thematic roles.
8. Re-rank documents based on the set of features that we compute, including answer type patterns. Some of the answer type patterns are based on the semantic labels.
9. Compute for each paragraph a confidence measure that it contains some relevant information. This includes N-Best count as one of the features.
10. Send tagged paragraphs that exceed a confidence threshold for extraction of the short answer required for TREC.

For the problem of question answering, we are more concerned with precision than recall, so we have to be careful in expanding the query words to get answers that are expressed in words quite different from the ones mentioned in the question. Semantic expansion will be performed when the system's confidence in the best candidate answer string without expansion is found to be below a certain threshold. Our mechanism for expansion is:

a. Submit original query words to IR engine and get back a rank-ordered set of documents.

b. Generate set of target words from top k documents based on the *idf* values of the words. We are experimenting with k in the range of 1–100.

c. Generate a set of target words from WordNet [8] synsets of original keywords.

d. Take the intersection of the two sets and add to the keyword set.

2.2 Features

Answer Identification We now discuss the features used for ranking the documents. The features are roughly ordered by decreasing salience.

▷ **Answer type** — In order to extract the answer to a question, the system needs to identify the expected answer type. Answers for short answer questions generally can be categorized as named entities and/or propositions. Summary information is often required for descriptive and definition questions. The system classifies the answer type by two features: 1) named entity class and 2) thematic role. Named entity class specifies one (or more) of 56 classes as the named entity class of the answer. We use 54 real named entity classes, one class representing the case where the answer is expected to be a named entity but one not known to the system, and one class for cases where the answer is not expected to be a named entity. The thematic role class identifies the thematic role in which the potential answer would tend to be found in the answering sentence.

▷ **Answer surface patterns** — Once the question type is known the next step is to identify candidate answers. One technique we use is based on generating a set of expected surface patterns for the answer. Sentences or snippets matching these patterns would get better scores than ones that did not. The patterns specify word and named entity-based regular expressions that are derived from a large corpus annotated with named entities; (simplified) examples include:

1. Some common question types, e.g., `[<PERSON_DESCRIPTION><PERSON_NAME>]` for questions like "Who is `<PERSON_NAME>`?"; `[<ORGANIZATION>, <CITY>, <STATE>, <COUNTRY>]` for questions asking about the location or address of an organization.
2. Likely re-phrasings of the question, e.g., `[<PERSON> invented <PRODUCT>]` for questions like "Who was the inventor of `<PRODUCT>`?"
3. Occurrence statistics of the pattern in the corpus, e.g., `[<PERSON> (<YEAR>-<YEAR>)]` for birth dates of people.

▷ **Named entities in answer** — In the case of questions that expect a specific named entity (including the unknown named entity type) as the answer, candidates that do not contain that named entity are penalized. In the case that the answer is expected to be an unknown named entity, then candidates that contain an untagged sequence of capitalized words (a strong indicator of unknown named entities) are preferred.

▷ **Presence of focus word** — The presence of the focus word is an important feature for the overall score. For our purposes, a focus word is a word in the question that, or its synonym, is very likely to appear in the sentence that contains the answer.

▷ **Thematic role patterns** — While surface patterns for answers can provide valuable information when a match is found, the specific nature of the patterns and the limited occurrences of the answer string within the reformulations obtainable from the question do not always guarantee a surface pattern match. We also provide a more general set of expected answer patterns based on thematic roles. We expect that these patterns will have higher coverage than the more specific surface patterns. This feature scores sentences based on the presence of expected thematic roles and named entities existing in specific thematic roles.

Thematic role patterns help identify false positive answer candidates and help extract the exact answer boundary from the string This can be illustrated with the example in Figure 1

Question: Who assassinated President McKinley?

Parse: [$_{role=agent}$ Who] [$_{target}$ *assassinated*] [$_{role=patient}$ [$_{ne=person\_description}$ **President**] [$_{ne=person}$ **McKinley**]]?

Keywords: assassinated President McKinley

Answer named entity (ne) Type: Person

Answer thematic role (role) Type: Agent of target synonymous with "assassinated"

Thematic role pattern: [$_{role=agent}$ [$_{ne=person}$ `ANSWER`] $\wedge$ [$_{target}$ `synonym_of(assassinated)`] $\wedge$ [$_{role=patient}$ [$_{ne=person}$ `reference_to(President McKinley)`]]

`This is one of possibly more than one patterns that will be applied to the answer candidates.`

False Positives:

`Note: The sentence number indicates the final rank of that sentence in the returns, without using thematic role patterns.`

1. In [$_{ne=date}$ **1904**], [$_{ne=person\_description}$ **President**] [$_{ne=person}$ **Theodore Roosevelt**], who had succeeded the [$_{target}$ *assassinated*] [$_{role=patient}$ [$_{ne=person}$ **William McKinley**]], was elected to a term in his own right as he defeated [$_{ne=person\_description}$ **Democrat**] [$_{ne=person}$ **Alton B. Parker**].

4. [$_{ne=person}$ **Hanna**]'s worst fears were realized when [$_{role=patient}$ [$_{ne=person\_description}$ **President**] [$_{ne=person}$ **William McKinley**]] was [$_{target}$ *assassinated*], but the country did rather well under TR's leadership anyway.

5. [$_{ne=person}$ **Roosevelt**] became president after [$_{role=patient}$ [$_{ne=person}$ **William McKinley**]] was [$_{target}$ *assassinated*] [$_{role=temporal}$ in [$_{ne=date}$ **1901**]] and served until [$_{ne=date}$ **1909**].

Correct Answer:

8. [$_{role=temporal}$ In [$_{ne=date}$ **1901**]], [$_{role=patient}$ [$_{ne=person\_description}$ **President**] [$_{ne=person}$ **William McKinley**]] was [$_{target}$ *shot*] [$_{role=agent}$ by [$_{ne=person\_description}$ **anarchist**] [$_{ne=person}$ **Leon Czolgosz**]] [$_{role=location}$ at the [$_{ne=event}$ **Pan-American Exposition**] in [$_{ne=us\_city}$ **Buffalo**] , [$_{ne=us\_state}$ **N.Y.**]] [$_{ne=person}$ **McKinley**] died [$_{ne=date}$ **eight days later**].

Figure 1: An example where thematic role patterns help constraint the correct answer among competing candidates. All the named entities, but only roles pertaining to the *target* predicate are marked in the sentences.

- ▷ **Okapi scoring** — This is the Okapi BM25[14] score assigned by the information retrieval engine to the paragraph extracted in the very beginning.
- ▷ **N-gram** — Another feature that we use is based on the length of the longest n-gram (sequence of contiguous words) in the candidate answer sentences after removing the stopwords from both the question and the answer.
- ▷ **Case match** — Documents with words in the same case as in the question tend to be more relevant than those that have different case, so the former are given a relatively higher score. This is because capitalization helps disambiguate named entities from common words (at least in carefully written queries).

Confidence Annotation Once the likely answer candidates (paragraphs or sentences) are extracted for a question we need to estimate the likelihood of those being *good* answers. To do this, we have a scheme that annotates each with

# wrong (W)	# unsupported (U)	# inexact (X)	# right (R)	CW Score	NIL Precision	NIL Recall
411	7	3	79	0.226	0 / 9 = 0.000	0/46 = 0.00

Table 1: System A (Boulder) results; numbers are out of 500 questions.

some level of confidence.

We use a weighted linear combination of N-Best answer count, Named Entity class of the answer, and the N-gram length features to calculate the degree of confidence.

2.3 Current Implementation

In this section we discuss the state of the current implementation for system A, which was used to produce the first run submitted to the TREC 2002 question answering track.

TREC-2002 Database The text database comprises non-stemmed, case-folded indexing of the TREC/AQUAINT corpus using a modified version of the mg (Managing Gigabytes) search engine [15] that incorporates the Okapi BM25 ranking scheme [14] and an expanded set of characters forming an indexed unit—so as to accommodate hyphenated words, URLs, emails etc. Each indexed document is a collection of segmented sentences forming a paragraph in the original corpus.

Answer Identification We currently use a rule-based question classifier that identifies the named entity type and thematic role of the expected answer to each question. For each query, the top N (2500, based on 85% recall threshold) ranked paragraphs are retrieved for processing. A set of documents are carried over from the list of documents retrieved by the IR engine, using gradually diluted boolean filters, until there are no more keywords to drop, or the cumulation of filtered documents exceeds a threshold of n (currently set to 10). We call this the *boolean peel-off* strategy. The answer named entity type is used to filter out documents that do not contain the required named entity type.

Answer Extraction For questions in which the answer is a specific named entity or a thematic role, if the top ranking sentence contains only one instance of that element, that instance is returned as the answer. In many cases, however, there is more than one element of the predicted type. In such cases, the system selects the element with the shortest average distance from each of the query words present in that snippet. There are penalties added for some punctuation symbols like commas, semi-colons, hyphens etc. In cases when the required answer is not a known named entity, the answer extractor tries to find a sequence of capitalized words that could be a probable named entity. In case the expected answer is not a named entity, and the thematic role cannot be identified without much ambiguity, then the system tries to find an apposition near the question words, and extracts that as the answer. An example would be definition questions, of the style "What is X?". Failing to get any of the above, the system replies that it does not have an answer to that particular question in the given corpus.

Confidence Annotation The prior probability of correct answers for a particular question type, along with the n-best count, are used to assign a confidence measure. The system then additionally tests whether the candidate at rank two has more counts in the top ten candidates than the candidate at rank one. If so, it is promoted to rank one.

2.4 Results

The results of System A on the TREC-2002 test set are presented in Table 1. These results are consistent with what we expected based on our TREC 9 and 10 development test set. Note that we answered "no answer" or NIL only if no answer was returned by our standard algorithm, which was rarely.

3 System B — The Columbia System

3.1 Overview of System Operation

Our second foundation system, developed at Columbia University, uses a hybrid architecture that evolved from an initial version that relied solely on the web as a source of answers. We focused on query expansion and candidate-answer scoring in order to perform search in parallel over open (the web) and closed (TREC) collections and then combine the results. For example, as described in more detail in Section 3.5, we experimented with a strategy that interleaves information learned from the web search to re-query the TREC document set.

In our initial, web-based system we adopted two working assumptions: First, given the size and redundancy of the web, a database of paraphrase patterns where more specific patterns are prioritized would be both necessary and effective for finding relevant documents (i.e., we aimed at precision rather than recall). Second, scores for candidate answers would be a composite function of attributes of the query formulation process, and of the space of candidate answers. For example, more specific queries have higher default scores, and an answer string retrieved from multiple documents is weighted higher than an answer string that appears in relatively few documents. Finally, we assumed that the question of how best to capitalize on these two assumptions would be primarily an empirical one. Thus questions that are superficially similar might require rather distinct queries. For example, questions that contain a noun denoting a leadership position, as in "Who was the first commander of `FLEET X`?" and "Who was the first coach of `TEAM Y`?" might both benefit from query expansion rules in which the related verbs appear (e.g., "`<TERM>` commanded `FLEET X`" and "`<TERM>` coached `TEAM Y`"), but this would not be the case for the structurally similar question "Who was the first general of `FORCE Z`?".

The remainder of this section documents four key modules of System B's architecture. There is a pipeline of three modules responsible for distinct phases of the query expansion process: paraphrasing of patterns derived from the question string (Section 3.2 below); modification of queries, e.g., by inserting terms likely to occur with the answer ([1]; Section 3.3); and term prioritization (Section 3.4). At each phase of the query expansion process we modify according to the results of that phase scoring weights that guide how likely that version of the query is to lead to the correct answer. Finally, after a candidate answer set has been assembled, these weights are assembled into a single score (Section 3.5). If, as in the case of our second run submitted to TREC, candidate answer sets for a given question come from distinct document collections, the source collection is considered in ranking potential answers (Section 3.5).

We conclude this section with a brief discussion of the results obtained on the data and questions of the 2002 TREC Q&A track.

3.2 Question Paraphrasing

In order to generate queries providing high precision coverage of the answer space for a given question, custom rules were developed providing a mapping from a given question type to a set of paraphrasing patterns which would generate alternative queries. For example, the question string "Where is the Hudson River located?" may result in the generation of queries including "Hudson River is located in", "Hudson River is located", "Hudson River is in the", "Hudson River is near", and "Hudson River in". Since we often do not have specific information about the question target, our paraphrasing rules allow changes on articles (definite, indefinite, or no article), number, and function words to maximize our coverage of the collection.

A two-level scoring scheme was implemented for these queries whereby each was scored based on the specificity of its query string as well as that of the paraphrasing pattern that generated it. Specificity here is defined by the length of the query string (or the length of the shortest possible generated query string in the case of the paraphrasing patterns); longer queries typically return fewer results than shorter (more general) queries. These query scores are used to aid the scoring and subsequent ranking of the returned results along with other factors that rate the results themselves (see Section 3.5).

3.3 Query Modification

Knowing the type of the question can be very helpful not only for defining the type of the answer but also for better targeting of the search of the documents which might contain a potential answer. This process can be viewed as connecting the answer and question spaces [3]. A question like "What is the length of the Amazon?" produces no useful results among the top 20 when submitted to Google, because among the two content words, "length" is not likely to appear directly in the answer, and "Amazon" is highly ambiguous (the river, the mythological female warrior, the online company, to name just the more common senses). "Length" is a very good *question* word, but needs to be mapped to corresponding *answer* words such as "miles" and "kilometers". If we expand the query by adding either *km* or *mi*, the first hits returned by Google are about the Amazon river and contain the answer.

In our TREC 2002 system we used query expansion only for questions that required answers consisting of a number plus a measurement unit. All the questions that required a number as an answer were categorized under the general type NUMBER by our question classifier. We built a further classifier that translated some of the question words to subtypes of NUMBER, namely DISTANCE, WEIGHT, SPEED, TEMPERATURE, MONEY, and OTHER. The classifier was constructed by training using RIPPER [6] to produce a set of rules. For each of the questions classified into the above subtypes, the classifier returns an automatically compiled list of words expressing the appropriate measurement units, which were added to the query.

3.4 Query Prioritization

Query prioritization scores queries so that those with highest predicted answer precision (i.e., number of documents retrieved by a query that contain the correct answer(s) divided by the total number of documents retrieved) will be attempted first during document retrieval. This is because, first, our paraphrasing and modification mechanisms can generate thousands of queries and it is impractical to submit all of those to the search engine, and, second, because we can tell that some query rewriting mechanisms are more likely to be accurate than others.

For example, given the question "How many soldiers died in World War II?", our system will generate many queries, using the paraphrasing (Section 3.2) and modification (Section 3.3) rules mentioned above. The generated queries include "World War II" (q_a) and "soldiers" (q_b). We want query prioritization to assign q_a a higher score since it clearly is more specific and relevant to this question, and thus is likely to have a higher predicted answer precision.

Unlike previous TREC systems [7, 12], which mainly applied heuristics for query prioritization, we empirically built our query prioritization algorithm using statistics collected over previous TREC question-answer pairs. We analyzed the relation between a query term (see Sections 3.2 and 3.3 for query term generation) and its answer precision. We considered various features of a query term, including the term's syntactic role (e.g., noun phrase, verb phrase, and head noun), morphological features (e.g., upper case for proper noun), and inverse document frequency (IDF). We found that IDF consistently reflected answer precision.

The query scoring algorithm for non-paraphrase-based queries relies on two functions:

- ▷ The term-scoring function T maps terms to term scores, where terms are strings of one or more words which are part of a query. The term score of a multiword term X is the product of the IDF values of the individual words forming the term. A minimum IDF value of 1.05 is used in this case to ensure that even common terms slightly boost the score of a multiword term. In addition, a suitably high value is used for words with unknown IDF values. We use IDF values computed over a large body of recent general news text.

- ▷ The query-scoring function Q maps a set of one or more terms (a query) to a query score. The query score for query Y is the product of the term scores of the query's component terms; given the definitionof term scores above, this means that a query's score is the product of the IDF values for all words in all phrases of the query.

As mentioned above, the query scores produced are used in document retrieval, answer selection, and answer ordering (for listing first the answers to questions where our system has the greatest confidence). For the latter purpose, scores must be normalized. This is because answer ordering requires us to compare answer confidence across answers to different questions; since part of our confidence is determined by the score of the query leading to the answer (see Section 3.5), we must be able to make a meaningful comparison between query scores for queries produced for two different questions. Therefore, scores for each set of queries produced for a given question are normalized using division by the highest query score for a query generated for that question. Thus, the highest scoring query produced for any question will have a score of 1, while lower scoring queries will have scores between 0 and 1.

It is important to note that our system scores any paraphrase-based query above any keyword-based query, irrespective of the functions mentioned above.[1] This is because the paraphrasing rules (Section 3.2) were hand-crafted and produced with an eye toward high precision. Thus, the scores produced by this module of the system were only used to differentiate between keyword-based queries.

3.5 Combining Evidence from Multiple Sources

Even when the answer must be found in a specific collection, as in the TREC evaluation, there are benefits in combining evidence from multiple collections: the answer can be found in a larger or broader collection, in which case the smaller collection can be searched again with that specific answer in mind; and, if no answer can be found in the smaller collection, the confidence in the answer from the larger collection can help determine whether "no answer present in the collection" (NIL) should be returned.

To this end, we have designed a general mechanism using *wrappers* that interface our system to different collections and/or search engines using a common API. We have implemented wrappers for the Google, Glimpse [13], and mg [15] search engines, and for TREC-11 we ran system B using a combination of Google on the web (a broad collection) and mg on the TREC/AQUAINT collection (the collection where the answer must be found).

Our system returns a list of answers extracted from each source (search engine and collection combination), which may include the same string multiple times. Each instance of an answer has an associated confidence score, which depends on the query used to retrieve the answer (see Sections 3.2 and 3.4), the confidence score returned by the search engine (not implemented for our TREC experiments), and the confidence of the answer extractor in selecting an appropriate phrase from the returned document. These instance scores are combined for each potential answer (i.e., across all instances where the same string is returned) using the following formula for computing the cumulative score after n instances of the same string have been processed

$$\text{cumulative\_score}_n = 1 - [(1 - \text{cumulative\_score}_{n-1}) \times (1 - \text{instance\_score}_n)]$$

which ensures that answers occurring multiple times are weighted higher, taking into account the evidence for them each time they are found.

An answer may be found in one or more sources. We employ the following algorithm for calculating a composite score based on the cumulative scores of each string proposed as a potential answer, from the web or the TREC collection. The algorithm distinguishes the following cases:

- ▷ No answers found in either collection. Since this probably represents a system failure (it is unlikely that a TREC question would not be answerable from both the web and the TREC collection), we return NIL with zero confidence.

- ▷ One or more answers found in the TREC collection, but no answers found from the Web. We return the best answer extracted from the TREC collection but depreciate its confidence by a constant factor because the coverage of the web would make it unlikely that no answers at all could be found there while the TREC collection would be able to provide one.

[1] Queries augmented with the query modification techniques discussed in Section 3.3 are considered keyword-based queries in this context.

- ▷ No answers found in the TREC collection, but one or more answers found from the Web. This is our *prima facie* case for a NIL answer; however, since often we got our answer from the web using a modified query, we re-query the TREC collection using the answer identified from the web. If that step succeeds, we report the answer with a combined confidence as in the next case below; otherwise we report NIL, with confidence equal to our confidence in the web answer.

- ▷ One or more answers found in each collection, and both top ranked answers are the same. We report that answer, with reinforced confidence according to the formula

$$\text{combined\_confidence} = 1 - [(1 - \text{confidence\_TREC}) \times (1 - \text{confidence\_web})]$$

- ▷ One or more answers found in each collection, but the top ranked answers are different. We report the TREC answer, but reduce its confidence by the formula

$$\text{combined\_confidence} = \text{confidence\_TREC} \times (1 - \text{confidence\_web})$$

3.6 Results

For the 500 questions in the TREC-11 question, System B produced 58 correct answers, 8 unsupported, 2 inexact, and 432 wrong. The system's unweighted precision is 11.6%, while its confidence-weighted score is 0.178, representing a significant boost over the unweighted precision. We produced a lot of NIL answers (195, or 39% of our total answers), which indicates that we were too conservative in failing to choose low-confidence answers found in the TREC collection. We did retrieve a third of the NIL answers present in the test set, but overall System B performed less well than System A which obtained a confidence-weighted score of 0.226 while producing very few NILs (1.8% of its total answers). On non-NIL answers, the two systems performance is closer (unweighted precision of 16.1% for system A and 14.1% for system B).

We attribute the lower performance of System B to two additional factors beyond the excessive production of NILs: First, we used a very simple extractor for selecting the answer out of the sentence where it was found, and the extractor failed to produce the correct phrase in a number of cases where we succeeded in finding the appropriate sentence. Second, our question classifier was not always successful in predicting the correct question type, producing no label for 77 (15.4%) of the questions. We performed significantly worse on those questions than in *who*, *when*, *where*, or *what* questions with well-extracted types.

4 Conclusion and Future Work

We have described two systems that handle questions with short, factual answers. These systems were developed in a very brief time frame, and are our first entry in the TREC Q&A track. They represent for us a starting point for an overall question answering architecture, to which we are adding additional capabilities. In the months since TREC, we have worked on developing detailed XML-based protocols for communication between modules in a common architecture. The system we are building by combining elements of the two systems discussed in this paper utilizes the best-performing components of each of the current systems. The modular architecture allows us to connect additional modules, and actual question answering is done in a distributed fashion, with most of question analysis done at Colorado and most of answer synthesis done at Columbia. Modules communicate with a central server via HTTP, while the architecture offers several communication interfaces at different levels of complexity (for example, one module may request only the high-level question type, while another may examine in detail the semantic parse). A web-based client provides a front end to the integrated system, allowing users in different locations to access the system.

Our research is moving towards questions with more complex answers, including opinions, events, definitions, and biographies. We recently completed a prototype module for processing definitional questions, and we are currently adding its functionality to our integrated system, so that for questions of the form "What is X?" we produce both a short, TREC-like, answer and an answer spanning several paragraphs. At the same time, we are continuing to tune individual components to enhance interoperability between the two sites; for example, we recently started re-focusing the semantic analysis (done at Colorado) on types of phrases that are likely to impact the processing of opinions and biographies (done at Columbia).

Acknowledgments

This work, as well as our broader research on question answering, is supported by ARDA AQUAINT contract MDA908-02-C-0008. We would like to thank Ralph Weischedel of BBN Systems and Technologies for giving us the named entity tagger (Identifinder), and Daniel Gildea for the FrameNet parser.

References

[1] Agichtein, E., Lawrence, S., and Gravano, L. "Learning Search Engine Specific Query Transformations for Question Answering". In *Proc. of the 10th International World-Wide Web Conference (WWW10)*, 2001.

[2] Baker, C., Fillmore, C., and Lowe, J. "The Berkeley FrameNet Project". In *Proceedings of the COLING-ACL*, Montreal, Canada, 1998.

[3] Berger, A., Caruana, R., Cohn, D., Freitag, D., and Mittal, V. "Bridging the Lexical Chasm: Statistical Approaches to Answer-Finding". In *Proceedings of the 23th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece, July 2000.

[4] Bikel, D., Schwartz, R., and Weischedel, R.. "An Algorithm that Learns What's in a Name". *Machine Learning*, 34:211–231, 1999.

[5] Brin, S. and Page, L. "The Anatomy of a Large-Scale Hypertextual Web Search Engine". *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[6] Cohen, W. W. "Fast Effective Rule Induction". In *Proceedings of the Twelfth International Machine Learning Conference (ML-95)*, 1995.

[7] Dumais, S., Banko, M., Brill, E., Lin, J., and Ng A. "Web question answering: Is more always better?". In *Proceedings of SIGIR-02*, pp. 291–298, 2002.

[8] Fellbaum, C., *editor*, "WordNet: An Electronic Lexical Database", MIT Press, 1998.

[9] Gildea, D. and Jurafsky, D. "Automatic Labeling of Semantic Roles". Technical Report TR-01-005, International Computer Science Institute, Berkeley, 2001.

[10] Harabagiu, S., Moldovan, D., *et al.* "Answering complex, list and context questions with LCC's Question-Answering Server". In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, pp. 355–361, Gaithersburg, Maryland, November 13-16, 2001.

[11] Hovy, E. and Hermjakob, U. "The Use of External Knowledge of Factoid QA". In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, pp. 644–652, Gaithersburg, Maryland, November 13-16, 2001.

[12] Lee, G. G., Seo, J., Lee, S., Jung, H., Cho, B. H., Lee, C., Kwak, B. K., Cha, J., Kim, D., An, J., Kim, H., and Kim, K. "SiteQ: Engineering High Performance QA System Using Lexico-Semantic Pattern Matching and Shallow NLP". In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, Gaithersburg, Maryland, November 13-16, 2001.

[13] Manber, U. and Wu, S. "Glimpse: A tool to search through entire file systems". In *Proceedings of the Winter USENIX Conference*, January 1994.

[14] Robertson, S. and Walker, S. "Okapi/Keenbow at TREC-8". In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, pp. 151–162, Gaithersburg, Maryland, November 17-19, 1999.

[15] Witten, I., Moffat, A., and Bell, T. "Managing Gigabytes: Compressing and Indexing Documents and Images", Morgan Kaufmann Publishing, San Francisco, May 1999.

Building an Arabic Stemmer for Information Retrieval

Aitao Chen
School of Information Management and Systems
University of California at Berkeley, CA 94720-4600, USA
aitao@sims.berkeley.edu

Fredric Gey
UC Data Archive & Technical Assistance (UC DATA)
University of California at Berkeley, CA 94720-5100, USA
gey@ucdata.berkeley.edu

1 Summary

In TREC 2002 the Berkeley group participated only in the English-Arabic cross-language retrieval (CLIR) track. One Arabic monolingual run and three English-Arabic cross-language runs were submitted. Our approach to the cross-language retrieval was to translate the English topics into Arabic using online English-Arabic machine translation systems. The four official runs are named as BKYMON, BKYCL1, BKYCL2, and BKYCL3. The BKYMON is the Arabic monolingual run, and the other three runs are English-to-Arabic cross-language runs. This paper reports on the construction of an Arabic stoplist and two Arabic stemmers, and the experiments on Arabic monolingual retrieval, English-to-Arabic cross-language retrieval.

2 Background

Arabic has much richer morphology than English. Arabic has two genders, *feminine* and *masculine*; three numbers, *singular*, *dual*, and *plural*; and three grammatical cases, *nominative*, *genitive*, and *accusative*. A noun has the nominative case when it is a subject; accusative when it is the object of a verb; and genitive when it is the object of a preposition. The form of an Arabic noun is determined by its gender, number, and grammatical case. The definitive nouns are formed by attaching the Arabic article ال to the immediate front of the nouns. As an example, the Arabic word الطالبة means *the student* (feminine). Sometimes a preposition, such as ب (by) and ل (to), is attached to the front of a noun, often in front of the definitive article. For example, the Arabic word للطّالبين means *to the students* (masculine). Besides prefixes, a noun can also carry a suffix which is often a possessive pronoun. For example, the Arabic word بطالبي (by my student) can be analyzed as ي + طالب + ب, with one prefix ب (by) and one pronoun suffix ي (my). In Arabic, the conjunction word و (and) is often attached to the following word. For example, the word وبطالبها means *and by her student* (masculine). Arabic has two kinds of plurals: *sound* plurals and *broken* plurals. The *sound* plurals are formed by adding plural suffixes to singular nouns. The plural suffix is ات for feminine nouns in all three grammatical cases, ون for masculine nouns in nominative case, and ين for masculine nouns in genitive and accusative cases. For example, the word مدرّسون (teachers, masculine) is the plural form of مدرّس (teacher, masculine) in nominative case, and مدرّسين (teachers, masculine) is the plural form of مدرّس (teacher, masculine) in genitive or accusative case. The plural form of مدرّسة (teacher, feminine) is مدرّسات (teachers, feminine) in all three grammatical cases. The dual suffix is ات for the nominative case, and ين for the genitive or accusative. The word مدرّسان means *two teachers*. The formation of broken plurals is more complex and often irregular; it is, therefore, difficult to predict. Furthermore, broken plurals are very common in Arabic. For example, the plural form of the noun طفل (child) is

أطفال (children), which is formed by attaching the prefix أ and inserting the infix ا. The plural form of the noun كتاب (book) is كتب (books), which is formed by deleting the infix ا. The plural form of امرأة (woman) is نساء (women). The plural form and the singular form are almost completely different. The examples presented in this secion show that an Arabic noun could potentially have a large number of variants, and some of the variants can be complex because of the prefixes, suffixes, and infixes. As an example, the word ولأطفالها (and to her children) can be analyzed as و + ل + أطفال + ها. It has two prefixes and one suffix.

Like nouns, an Arabic adjective can also have many variants. When an adjective modifies a noun in a noun phrase, the adjective agrees with the noun in gender, number, case, and definiteness. An adjective has a masculine singular form such as جديد (new), a feminine singular form such as جديدة (new), a masculine plural form such as جلّد (new), and a feminine plural form such as جديدات (new). For example, المدرّس الجديد means *the new teacher* (masculine), and المدرّسون الجلّد means *the new teachers* (masculine). The adjective has the feminine singular form when the plural noun denotes something inanimate. As an example, the word جديدة (new) in الكتب الجديدة (the new books) is the feminine singular form.

Arabic verbs have two tenses: perfect and imperfect. Perfect tense denotes actions completed, while imperfect denotes incompleted actions. The imperfect tense has four mood: indicative, subjective, jussive, and imperative [4]. Arabic verbs in perfect tense consist of a stem and a subject marker. The subject marker indicates the person, gender, and number of the subject. The form of a verb in perfect tense can have subject marker and pronoun suffix. The form of a subject-marker is determined together by the person, gender, and number of the subject. Take درس (to study) as an example, the perfect tense is درست for the third person, feminine, singular subject, درسوا for the third person, masculine, plural subject. A verb with subject marker and pronoun suffix can be a complete sentence. For example, the word درسته has a third-person, feminine, singular subject-marker ت (she) and a pronoun suffix ه (him), it is also a complete sentence, meaning "she studied him." Often the subject-makers are suffixes, but sometimes a subject-marker can be a combination of a prefix and a suffix. For example, the word study in a negative sentence is تدرسي (did not study). For verbs in imperfect tense, in addition to the subject-marker, a verb can also have a mood-marker.

3 Test Collection

The document collection used in TREC 2002 cross-language track consists of 383,872 Arabic articles from the Agence France Press (AFP) Arabic Newswire during the period from 13 May, 1994 to 20 December, 2000. There are 50 English topics with Arabic translations. A topic has three tagged fields: *title*, *description*, and *narrative*. The newswire articles are encoded in Unicode (UTF-8) format, while the topics are encoded in ASMO 708.

4 Preprocessing

Because the texts in the documents and topics are encoded in different schemes, we converted both the documents and topics to Windows CP-1256 encoding. The set of valid characters include the Arabic letters and the English letters in both lower and upper cases. The Arabic punctuation marks, ،, ؛, and ؟, were considered as delimiters. A consecutive sequence of valid characters was recognized as a word in the tokenization process. The words that are stopwords were removed during documents and topics indexing. We say a word is *minimally* normalized when ٲ, ٵ, ٳ, ٱ, أ, إ, and آ are changed to ا. A word is *lightly* normalized when additionally the Shadda character (the character above ل in لّ) is deleted, and the characters آ, أ, and إ are changed to ا, the final ى is changed to ي, and the final ه is changed to ة. In the Arabic document collection, the word امرأة (woman) is sometimes spelled as أمرأة or إمرأة. The Arabic shadda character is sometimes dropped in spelling. For example, for the word مدرّس (teacher) is sometimes spelled as مدرس.

5 Construction of stopword list

At TREC 2001, we created an Arabic stopword list consisting of Arabic pronouns, prepositions, and the like that are found in an elementary Arabic textbook [4] and the Arabic words translated from an English stopword list. For TREC 2002, we first collected all the Arabic words found in the Arabic document collection. The number of unique Arabic words found in the collection after minimal normalization is 541,681. We then translated the Arabic words, word-by-word, into English using the *Ajeeb* online English-Arabic machine translation system available at http://www.ajeeb.com. From this Arabic-English bilingual wordlist, we created an Arabic stopword list consisting of the Arabic words whose translations consists of only English stopwords. The Arabic stopword list has 3,447 words after minimal normalization, containing stopwords such as لكنكم (you), فية (in him), بينهم (between them), and بعدما (after). The English stopword list has 360 words. There are a couple of reasons why the Arabic stopword list automatically generated is much larger than the English stopword list. First, pronouns can have more than one form. For example, the Arabic word for *these* has four forms: هاتان (feminine, nominative), هاتين (feminine, genitive/accusative), هذان (masculine, nominative), and هذين (masculine, genitive/accusative). Second, pronouns and prepositions are sometimes joined together.

6 Construction of stemmers

At TREC 2001, we built a rather simple Arabic stemmer to remove from words the definite article prefix ال, the plural suffixes ان, ون, and ات, and the suffix ة. At TREC 2002, we created two Arabic stemmers, a *MT-based stemmer* and a *light stemmer*.

6.1 MT-based stemmer

We built a MT-based Arabic stemmer from the Arabic words found in the Arabic documents and their English translations using the online *Ajeeb* machine translation system. We partitioned the Arabic words into clusters based on the English translations of the Arabic words. The Arabic words whose English translations, after removing English stopwords, are conflated to the same English stem form one cluster. And all the Arabic words in the same cluster are conflated to the same Arabic word, the shortest Arabic word in the cluster. For example, an English stemmer usually changes plural nouns into singular, so *children* is changed to *child*. In order to change the variants of the Arabic word for *child* or *children* to the same Arabic stem, we first grouped all the Arabic words whose English translations contain the headword *child* or *children*. Then in stemming, all the Arabic words in this group are changed to the shortest Arabic word in the group. The Arabic adjectives and verbs were stemmed in the same way. For English, we used a morphological analyzer [2] to map plural nouns into singular form, verbs into the infinitive form, and adjectives into the positive form. This stemmer changes the *broken* plural forms of an Arabic word into its singular form. The broken plural forms are common and irregular, so it is generally difficult to write a stemmer to change the broken plural forms to singular forms. For example, Table 1 presents part of the Arabic words whose English translations contain the headword *child* or *children*. All the Arabic words shown in table 1 belong to the same cluster since, after removing the English stopwords, the English translations consist of either the word *child* or *children*, both being conflated to the same word by the English morphological analyzer. In stemming, the Arabic words shown in table 1 are conflated into the same word طفل. The English translations were produced using the online Ajeeb machine translation system. One can also create an Arabic stemmer from English/Arabic parallel texts or bilingual dictionaries. With a large English/Arabic parallel corpus available, one can first align the texts at the sentence level, then use a statistical machine translation toolkit such as GIZA++ to create an Arabic-to-English translation table. If we keep only the most likely English translation for an Arabic word, then we have a bilingual wordlist. Using this bilingual wordlist, we can translate all the Arabic words found in the Arabic document collection into English. We can create an Arabic stemmer by partitioning the Arabic words into clusters, each consisting of the Arabic words whose English translations are conflated to the same word by the English morphological analyzer. Stemmers for other languages can also be automatically generated using this method as long as some translingual resources, such as MT, parallel texts, or bilingual dictionaries, are available.

Arabic word	English translation	Arabic word	English translation	Arabic word	English translation	Arabic word	English translation
أطفال	children	اطفالهن	their children	بطفل	by child	فالطفلة	then the child
أطفالا	children	اطفالي	my children	بطفلة	by child	فطفل	then child
أطفالنا	our children	الأطفال	children	بطفلتنا	by our child	كأطفال	as children
أطفاله	and his children	الاطفال	children	بطفلته	by his child	كالطفل	as the child
أطفاله	his children	الطفل	the child	بطفله	by his child	لأطفال	to children
أطفالها	her children	الطفلان	the children	بطفلها	by her child	لطفلها	to her child
أطفالهم	their children	الطفلة	the child	بطفلهما	by their child	للّطفلة	to the child
أطفالهن	their children	الطفلتان	the children	بطفلين	by children	وأطفالنا	and our children
أطفالي	my children	الطفلتين	the children	بطفليها	by her children	والأطفال	and the children
اطفال	children	الطفله	the child	طفل	child	و بطفل	and by child
اطفالا	children	الطفلين	the children	طفلا	child	و بطفلين	and by children
اطفالك	your children	بأطفال	by children	طفلان	children	وطفلة	and child
اطفالكم	your children	بأطفاله	by his children	طفلاها	her children	وطفلتان	and children
اطفالكن	your children	بأطفالها	by her children	طفلة	child	وطفلنا	and our child
اطفالنا	our children	بالأطفال	by the children	طفلت	child	وطفلها	and her child
اطفاله	his children	بالطفل	by the child	طفلتان	children	وطفليه	and his children
اطفالها	her children	بالطفلة	by the child	طفلتة	his child	وطفليها	and her children
اطفالهم	their children	بالطفلتين	by the children	طفلتنا	our child	ولأطفالها	and to her children
اطفالهما	their children	بالطفلين	by the children	طفلته	his child	وللّطفل	and to the child

Table 1: Arabic words whose English translations contain the headword *child* or *children*.

6.2 Light stemmer

We developed a second Arabic stemmer called *light stemmer* that removes only prefixes and suffixes. We identified one set of prefixes and one set of suffixes that should be removed based on the grammatical functions of the affixes, their occurrence frequencies among the Arabic words found in the Arabic document collection, the English translations of the affixes, and empirical evaluation using the test collection of the previous CLIR track. We generated three lists consisting of the initial, the first two, or the first three characters, respectively, of the Arabic words in the document collection, and three lists consisting of the final, the last two, or the last three characters, respectively, of the Arabic words. We then sorted the six lists of suffixes or prefixes in descending order by the number of unique words in which a prefix or suffix occurs. Table 2 presents the most frequent one-, two-, and three-character prefixes among the unique Arabic words found in the document collection. The frequency shown in the table is the number of unique Arabic words that begins with a specific prefix. Table 3 shows the most frequent one-, two-, and three-character suffixes among the unique Arabic words. The frequency count for a given suffix is the number of unique Arabic words that end with that suffix. We identified 9 three-character, 14 two-character, and 3 one-character prefixes that should be removed in stemming, and 18 two-character, and 4 one-character suffixes that should be removed in stemming. The 9 three-character prefixes are وال (and the), بال (by the), فال (then the), كال (as the), ولّل (and to the), مال, الل, سال, لال. The 14 two-character prefixes to be removed are the most frequent ones as shown in table 2. Our light stemmer shares many of the prefixes and suffixes that should be removed with the light stemmer developed by Larkey et al. [5] and the light stemmer developed by Darwish[3].

The stemmer non-recursively removes the prefixes in the pre-defined set of prefixes, and recursively removes the suffixes in the pre-defined set of suffixes in the following sequence.

1. If the word is at least five-character long, remove the first three characters if they are one of the following: وال,

Rank	Initial character	Frequency	Initial two characters	Frequency	Initial three characters	Frequency
1	و	117324	ال	55364	وال	19411
2	ا	94043	وا	32787	الم	12711
3	ب	49319	با	16789	بال	9079
4	ل	48862	للّ	10912	الا	6666
5	م	33776	وم	10124	الت	3907
6	ت	25649	وت	9196	الب	2813
7	س	23385	وب	8865	وبا	2760
8	ف	21828	لا	7482	است	2559
9	ك	19794	سي	7447	وسي	2372
10	ي	19004	وس	7155	الس	2260
11	ن	10905	وي	6772	فال	2213
12	د	8445	ول	6527	الع	1973
13	ر	8345	كا	6083	للّم	1919
14	ش	7058	فا	5648	الك	1915
15	غ	6680	او	4933	الف	1783
16	ه	6435	ما	4877	كال	1751
17	ج	6383	لي	4749	للّا	1736
18	أ	5394	بو	4702	الن	1665
19	ع	5207	كو	4583	واس	1613
20	ح	4450	ان	4415	الح	1610
28					وللّ	1391
168					مال	412
203					اال	365
262					سال	312
268					لال	306

Table 2: Most frequent initial character strings.

بال, فال, كال, وللّ, مال, اال, سال, لال.

2. If the word is at least four-character long, remove the first two characters if they are one of the following: ال, وا, با, للّ, وم, وت, وب, لا, سي, وس, وي, ول, كا, فا.

3. If the word is at least four-character long and begins with و, remove the initial letter و.

4. If the word is at least four-character long and begins with either ب or ل, remove ب or ل only if, after removing the initial character, the resultant word is present in the Arabic document collection.

5. Recursively strips the following two-character suffixes in the order of presentation if the word is at least four-character long before removing a suffix: ها, ية, هم, نا, ما, وا, يا, ني, يا, هن, كم, كن, تم, تن, ين, ان, ات, ون.

6. Recursively strips the following one-character suffixes in the order of presentation if the character is at least three-character long before removing a suffix: ة, ه, ي, ت.

Rank	Final character	Frequency	Last two characters	Frequency	Last three characters	Frequency
1	ا	91571	ها	26412	تها	6544
2	ن	69574	ين	24601	هما	6286
3	ي	52418	ان	19089	تين	4591
4	ة	44683	ات	17612	لين	4262
5	ه	34288	ون	15724	تهم	3836
6	ت	33351	ية	13877	يان	2960
7	م	27346	هم	13570	يتش	2747
8	ر	25748	نا	11794	تان	2722
9	و	21123	ما	8811	اني	2534
10	ل	18531	وا	8276	يون	2443
11	س	14668	يا	7702	رات	2250
12	د	13352	ني	7553	مان	2056
13	ى	12037	ته	7379	اته	2050
14	ف	11265	ري	5187	تنا	1953
15	ك	9278	لي	5090	رين	1918
16	ب	8863	ار	5027	نية	1833
17	ز	6973	ير	4869	رها	1805
18	ش	6777	يه	4611	نها	1801
19	ش	6777	تي	4377	ينا	1775
20	ع	5987	يل	4268	لها	1759

Table 3: Most frequent last character strings.

In our implementation, the suffix ي is removed only if the word is at least four-character long and the resultant word after removing the suffix is present in the Arabic document collection. The prefix وبال is often the combination of three prefixes و (and), ب (by), and ال (the), and should be removed. The light stemmer we used for the TREC 2002 experiments did not remove this prefix combination. We decided to remove the initial letter WAW (و) since it the most frequent initial letter and often is the conjunction word attached to the following word. The other two initial letters that were removed are BEH (ب) and LAM (ل). The prefix ب is sometimes a preposition prefix, meaning *by*, and the prefix ل is also sometimes a preposition prefix, meaning *to*. Our light stemmer removes ب and ل only when, after removing the prefix, the resultant stem is also a word in the collection.

Among the two-letter suffixes to be removed, six are pronoun suffixes (ها, هم, نا, هن, كم, كن); four are plural suffixes (ين, ان, ات, ون); and three are subject markers (وا, تم, تن). The suffix ية is a nisba ending. The single-letter suffix ة is the feminine ending, ه a pronoun suffix, ي a pronoun suffix, and ت a subject marker. Sometimes the suffix ة is inseparable since, if removed, the resultant word is completely a different word. As an example, the word الملكة means *the queen*, after removing the suffix ة, the resultant word الملك means *the king*.

7 Experimental Results

7.1 Retrieval system

The retrieval system we used for the experiments is an implementation of the retrieval algorithm presented in [1]. For term selection, we assume the top-ranked m documents in the initial search are relevant, and the rest of the documents in the collection are irrelevant. For the terms in the documents that are presumed relevant, we compute term relevance weighting [6] as follows:

$$w_t = log\frac{m_t(n - n_t - m + m_t)}{(m - m_t)(n_t - m_t)} \tag{1}$$

where n is the number of documents in the collection, m the number of top-ranked documents after the initial search that are presumed relevant, m_t the number of documents among the m top-ranked documents that contain the term t, and n_t the number of documents in the collection that contain the term t. Then all the terms found in the top-ranked m documents are ranked in decreasing order by relevance weight w_t. The top-ranked k terms are weighted and then merged with the initial query terms to create a new query. Some of the selected terms may be in the initial query. For the selected top-ranked terms that are not in the initial query, the weight is set to 0.5. For those top-ranked terms that are in the initial query, the weight is set to $0.5*t_i$, where t_i is the occurrence frequency of term t in the initial query. The selected terms are merged with the initial query to formulate an expanded query. When a selected term is one of the query terms in the initial query, its weight in the expanded query is the sum of its weight in the initial query and its weight assigned in the term selection process. For a selected term that is not in the initial query, its weight in the final query is the same as the weight assigned in the term selection process, which is 0.5. The weights for the initial query terms that are not in the list of selected terms remain unchanged.

A query, like a document, is normally represented in our retrieval system by a set of unique words in the query with within-query term frequency. For the experiments reported in this paper, a word occurring n times in a query is represented by n occurrences of the same word with within-query frequency of one.

7.2 Monolingual Retrieval Results

The BKYMON run is our only official Arabic monolingual run in which only the *title* and *desc* fields in the topics were indexed. After removing stopwords from both documents and topics, the remaining words were stemmed using Berkeley light stemmer as described in section 6.2. The stopword list used in this run was the one created from the translations of Arabic document words using the online Ajeeb machine translation. The development of the Arabic stoplist was described in section 5. The stopword list has 2,942 words after light normalization. Table 4 presents the evaluation results for additional retrieval runs.

The monolingual run *mon0* was produced without stemming. The words were lightly normalized and stopwords removed. Two runs were performed using overlapping trigram indexing, one without word boundary crossing (mon1) and the other with word boundary crossing (mon2) . For example, without word boundary crossing, the following trigrams are produced from the phrase صيانة بلوغا: صيا,يان,انة, بلو, لوغ, وغا. But with word boundary crossing, two additional trigrams, نةب and ةبل, are produced. The words were lightly normalized and the stopwords were removed before trigrams were generated from the normalized words.

The monolingual run *mon3* used the light stemmer named *Al-Stem*, developed by Darwish [3]. The numeric digits from '0' to '9' are treated as part of a token in Darwish's stemmer which also reduces 616 unnormalized words found in the Arabic documents to empty string, effectively treating them as stopwords. The stemmer also normalizes words. For the run *mon3*, words were aggressively normalized within the stemmer. For all other runs, the numeric digits were treated as word delimiters, and the words were normalized using our own light normalizer.

For the run *mon4*, the words were stemmed using the automatically generated MT-based stemmer. The words were first normalized and then the stopwords removed.

For the runs, mon0, mon3, mon4, and BKYMON, 20 words were selected from the top-ranked 10 documents for query expansion; and for the runs, mon1 and mon2, 40 trigrams were selected from the top-ranked 10 documents for query expansion.

The increase in performance without query expansion is substantial, however, the difference remains small after query expansion.

7.3 Cross-language Retrieval Results

Our approach to cross-language retrieval was to translate the English topics into Arabic, and then search the translated Arabic topics against the Arabic documents. The source English topics were translated into Arabic using two online English-Arabic machine translation systems: *Ajeeb* and *Almisbar*, available at http://www.almisbar.com/.

			without expansion		with expansion	
run id	stemmer	index unit	recall	precision	recall	precision
mon0	NONE	word	4035	0.2365	4583	0.2872
mon1	NONE	trigram (without crossing)	3914	0.2398	4632	0.3239
mon2	NONE	trigram (with crossing)	4018	0.2479	4681	0.3178
mon3	Al-Stem stemmer	word	4500	0.2858	4864	0.3482
mon4	MT-based stemmer	word	4402	0.2948	4885	0.3348
BKYMON	Berkeley light stemmer	word	4543	0.3099	4952	0.3666

Table 4: Monolingual retrieval performances. The number of relevant documents for all 50 topics is 5909. Only the *title* and *description* fields were indexed.

We submitted three official cross-language runs: BKYCL1, BKYCL2, and BKYCL3. The BKYCL1 run was produced by merging the results of two English-to-Arabic retrieval runs: cl1 and cl2. The first run used the *Ajeeb* English-to-Arabic translations, and the second run used the *Almisbar* English-to-Arabic translations. For both intermediate runs, the words were stemmed using Berkeley's light stemmer after removing stopwords. For query expansion, 20 terms were selected from the top-ranked 10 documents. When two runs were merged topic by topic, the estimated probabilities of relevance were summed for the same documents. The merged list of documents was sorted by the combined estimated score of relevance, and the top-ranked 1000 documents per topic were kept to produce the official run BKYCL1. Only the *title* and *desc* fields in the topics were used to produce the BKYCL1 run. The average precision for run cl2 is 0.2782 with overall recall of 4823/5909. The average precision for run cl1 is 0.2962 with overall recall of 4441/5909.

The BKYCL2 run was produced by merging the results of three English-to-Arabic retrieval runs. The first two intermediate runs, cl1 and cl2, were the same two runs that were merged to produce BKYCL1 run. The third intermediate run, named cl3, was produced using the English-to-Arabic bilingual dictionary created from the U.N. English/Arabic parallel texts. The bilingual dictionary was provided as part of the standard translation resources for the cross-language track. Readers are referred to [7] for details on the construction of the bilingual dictionary. The English texts of the parallel corpus was stemmed using Porter stemmer, while the Arabic texts was stemmed using the Al-Stem stemmer which is part of the standard resources created for the cross-language track. Each entry in the English-to-Arabic bilingual dictionary consists of one stemmed English word and a list of stemmed Arabic words with the probabilities of translating the English word into the Arabic words. We translated the English topics into Arabic by looking up each English word after stemming using the same English porter stemmer in the English-to-Arabic bilingual dictionary, and keeping the two Arabic words of the highest translation probabilities. That is, the two most likely Arabic translations for each English word. Since only two Arabic translations were retained, the sum of their translation probabilities is at most one. In the case where the sum is less than one, the word translation probabilities were normalized so that the sum of the translation probabilities of the retained two Arabic words is one. The within-query term frequency of an English word is distributed to the retained Arabic words proportionally according their translation probabilities. For the cl3 run, we indexed the Arabic documents using the Al-Stem stemmer. The intermediate run cl3 was produced using the bilingual dictionary-translated topics. The average precision for run cl3 is 0.3072 with overall recall of 4826/5909. The official run BKYCL2 was produced by merging cl1, cl2, and cl3 runs. The estimated probabilities of relevance were summed during merging.

The official run BKYCL3 was produced again by merging two intermediate runs, cl3 and cl4. The cl3 run was described in the previous paragraph. The intermediate run cl4 was produced using the Ajeeb-translated topics like the cl1 run. The only difference is that the standard light stemmer, Al-Stem, was used in cl4. The average precision for run cl4 is 0.2710 with overall recall of 4350/5909.

The unofficial run, bkycl4, was produced like the official run BKYCL1 except that the MT-based stemmer was used here. The run bkycl4 was produced by merging cl5 and cl6. The cl5 run used the Ajeeb topic translations, while the cl6 run used the Almisbar topic translations. For both runs, the MT-based stemmer automatically constructed from Ajeeb-translated words was used. The average precision for run cl5 is 0.2733 with overall recall of 4118/5909, and the average precision for run cl6 is 0.2751 with overall recall of 4735/5909.

Table 5 shows the overall precision for the five runs. There are a total of 5,909 relevant documents for all 50 topics. The run BKYCL3 used standard resources only. Like the monolingual run, all cross-language runs were produced with query expansion in which 20 terms were selected from the top-ranked 10 documents after the initial search. Our best

Run ID	Type	Topic Fields	Recall	Precision	% of MONO
BKYMON	MONO	T,D	4952	0.3666	
BKYCL1	CLIR	T,D	4614	0.3000	81.83%
BKYCL2	CLIR	T,D	4874	0.3224	87.94%
BKYCL3	CLIR	T,D	4856	0.3089	84.26%
brkcl4	CLIR	T,D	4553	0.2857	77.93%

Table 5: Performances of the CLIR runs.

cross-language performance is 87.94% of the monolingual performance.

8 Conclusions

In summary, we performed one Arabic monolingual run and three English-Arabic cross-language retrieval runs, all being automatic. We took the approach of translating queries into document language using two machine translation systems. Our best cross-language retrieval run achieved 87.94% of the monolingual retrieval performance. We developed one MT-based Arabic stemmer and one light Arabic stemmer. The Berkeley light stemmer worked better than the automatically created MT-based stemmer. The experimental results show query expansion substantially improved the retrieval performance.

9 Acknowledgements

This research was supported by research grant number N66001-00-1-8911 (Mar 2000-Feb 2003) from the Defense Advanced Research Projects Agency (DARPA) Translingual Information Detection Extraction and Summarization (TIDES) program within the DARPA Information Technology Office.

References

[1] W. S. Cooper, A. Chen, and F. C. Gey. Full text retrieval based on probabilistic equations with coefficie nts fitted by logistic regression. In D. K. Harman, editor, *The Second Text REtrieval Conference (TREC-2)*, pages 57–66, March 1994.

[2] M. Zaidel D. Karp, Y. Schabes and D. Egedi. A freely available wide coverage morphological analyzer for english. In *Proceedings of COLING*, 1992.

[3] K. Darwish. *http://www.glue.umd.edu/~kareem/research/.*

[4] Peter F. Abboud [et al.], editor. *Elementary modern standard Arabic*. Cambridge University Press, 1983.

[5] L. Larkey, L. Ballesteros, and M.E. Connell. Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis. In *SIGIR'02, August 11-15, 2002, Tampere, Finland*, pages 275–282, 2002.

[6] S. E. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, pages 129–146, May–June 1976.

[7] Jinxi Xu, Alexander Fraser, and Ralph Weischedel. Trec 2001 cross-lingual retrieval at bbn. In E.M. Voorhees and D.K. Harman, editors, *The Tenth Text Retrieval Conference (TREC 2001)*, pages 68–77, May 2002.

Using Hierarchical Clustering and Summarisation Approaches for Web Retrieval: Glasgow at the TREC 2002 Interactive Track

Richard Osdin, Iadh Ounis and Ryen W. White

Department of Computing Science
University of Glasgow, Glasgow, G12 8QQ, Scotland
osdinra, ounis, ryen@dcs.gla.ac.uk

1. Introduction and Motivation

Current search engines are typified as having a lack of precision, coupled with an elongated ranked list style of result presentation. When combined, these factors make relevant data extraction increasingly complex. The main investigation of our participation in the Interactive Track of TREC 2002 is to assess the effectiveness of new visualisation techniques for displaying the results of search engines.

Our current system, provisionally named HuddleSearch, uses a newly developed clustering algorithm, which dynamically organises the relevant documents into a traversable hierarchy of general to more-specific cluster categories. We have extended our TREC-10 summarisation tool to also allow the summarisation of multiple documents; whereby a summary paints a caricature of the contents of a cluster, rather than an individual document, thus allowing the user to provisionally judge a cluster's relevance prior to viewing its contents. The interaction between the user and the system is further developed by the aid of an information visualisation tool. Our primary assumption is that the combination of both hierarchical clustering and summarisation tools will aid users in their interaction with the system in the Web context.

2. Systems

Our baseline system acts as a metasearch engine, providing a generic interface capable of displaying the results from any web search engine, simply by defining a wrapper specific to the baseline system. For the purposes of these experiments, we retrieve the results only from the provided Panoptic system[1], returning results solely from the .GOV domain. This system simply displays the title, a 200-character description and URL for each document, to provide the user with an element of familiarity with systems such as Google.

Our experimental system, HuddleSearch, extends the properties of our basic system, by enabling the user to find relevant documents quickly by means of navigating within a traversable hierarchy of clusters. When a user views a cluster title he or she gains an overview of the documents contained within it, and is then able to narrow in and view only the documents within a specified segmentation of choice, at a lower branch of the tree. In this way we address the problem of information overload; the user is able to reduce the relevant documents set by continually filtering out irrelevant documents in search of information satisfying their need.

Figure 1 shows the path between generality and specificity; where the retrieved set of documents contracts as the user progresses deeper into the cluster hierarchy. Unlike the flat clusters hierarchy of search engines like Vivisimo[2], WiseNut[3], or Grouper [2], HuddleSearch organises the clusters into a hierarchy, providing a better structure for the result set. Figure 2(a) presents a screenshot of the system when used as a metasearch engine on the Web. The clusters are shown as folders at the top of the interface. The title of the folder is indicative of the cluster content and the number on the folder represents cluster size.

We have complemented the conception of a cluster hierarchy by investigating the use of query-biased summarisation, previously explored by the Glasgow Information Retrieval Group, to provide short passages indicative of individual document content [1]. However, we have extended this practice to allow the summarisation of multiple documents. In a similar way to the previous work, the summaries are created on-the-fly at retrieval time, prior to the results page being displayed. Hereby, we introduce the creation of cluster summarisation, where groupings of significant sentences

[1] http://www.panopticsearch.com

[2] http://www.vivisimo.com

[3] http://www.wisenut.com

extracted from documents within a cluster are combined to produce a summary indicative of a cluster's content. The chosen sentences are ones that have a high degree of match with the user's query.

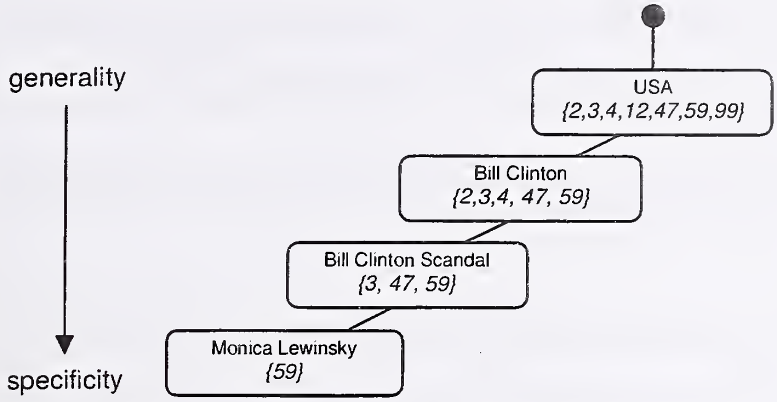
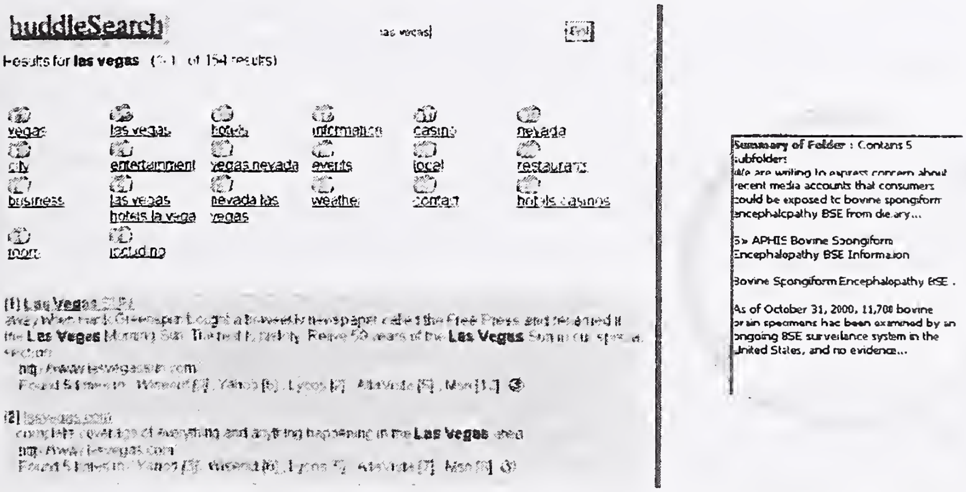


Figure 1: A dynamic hierarchical clustering approach

In addition to the traditional hyperlink method for navigating between clusters, as used by WiscNut, we have devised a visualisation tool, which allows the user to preview the contents of any given cluster with only the slightest mouse movement. This feature has been combined with our cluster summarisation; whereby a user can view the summary of any cluster with ease. The two complementary features enable the user to quickly glance at the contents of available clusters, initially assess relevance, and then select a cluster of interest. Searchers therefore do not waste valuable time viewing misleading document sets. Figure 2(b) displays our visualisation tool, which provides the user with a summary of a cluster's content when the mouse touches a cluster and appears on the display to the right of the clusters.



Results for **las vegas** (1-1 of 154 results)

[1] Las Vegas [illegible]
[illegible] the **Las Vegas** [illegible] **Las Vegas** [illegible]
[illegible]
Found 5 times in: [illegible] [illegible], Yahoo [6], Lycos [2], AltaVista [5], Msn [1]

[2] lasvegas.com
complete coverage of everything and anything happening in the **Las Vegas** area
Found 5 times in: Yahoo [3], [illegible] [6], Lycos [5], AltaVista [7], Msn [8]

Summary of Folder : Contains 5 subfolders
We are writing to express concern about recent media accounts that consumers could be exposed to bovine spongiform encephalopathy BSE from dietary...

5» APHIS Bovine Spongiform Encephalopathy BSE Information

Bovine Spongiform Encephalopathy BSE .

As of October 31, 2000, 11,700 bovine brain specimens had been examined by an ongoing BSE surveilance system in the United States, and no evidence...

Figure 2 (a): HuddleSearch interface (b): Cluster summary

3. Experiment

Two systems were used in our experiments: the provided Panoptic search engine, acting as a *baseline*, with its classical list-based approach, and the experimental system HuddleSearch, where the hierarchical clustering and the summarisation visualisation tool where activated. The HuddleSearch wrapper allows the Panoptic search engine results to be preserved[4], but masked the engine identity, hence avoiding any possible bias caused by previous searching experience. HuddleSearch and Panoptic were referred to only as System X and System Y respectively.

[4] Note that as mentioned in section 2, only the first 200-characters of each returned summary description is displayed.

A total of 16 users were recruited, each participant educated to at least a graduate level, from differing academic backgrounds, representative of the general web-using university populace.

Most users either work with or use computers for academic purposes frequently and have on average 5.7 years of web-searching experience. With the exception of just 1 user, Google was cited as being the search engine of choice.

Each user was required to carry out a total of 8 standard search tasks, equally split between the two systems. The tasks were allocated as required by the track guidelines to reduce potential learning effects and task bias. Figure 4 shows the tasks carried out by participants.

Following the track guidelines this year, each user was allowed a maximum of 10 minutes for each task. They were asked to use the system presented to them, and to perform the allocated task. All user actions were logged. Moreover, users were allowed to browse away from the result list to any degree. Due to the open nature of the collection this year, users were free to browse/save documents that were not in the TREC .GOV collection.

Government Regulation

- You are travelling from the Netherlands, and want to bring some typical food products as gifts for your friends. What are three kinds of food products from the Netherlands that you are not allowed to bring into the US? *(GR1)*
- You are concerned with privacy issues related to electronic information and would like to know what laws have been passed by the US Congress regarding these issues. Identify three such laws. *(GR2)*

Health or Project

- A friend has a private well which is the family's only source of drinking water. Locate a US publication, which contains guidelines for the maintenance of safe water standards for private well use. *(HP1)*
- You are not sure about the safety of genetically engineered foods, and would like to find more information and research on this topic. Name four potential types of safety problems that have been raised. *(HP2)*
- Name/find three research programs/projects that investigate the treatment/causes of dwarfism. *(HP3)*
- You are interested in learning more about what measures the US government has taken since 2001 to prevent Mad-Cow Disease. Identify three such measures. *(HP4)*

Travel

- You are planning a cycling expedition along the Silk Road in Central Asia. Find a website that is a good source information about health precautions should you take. *(T1)*
- You are planning to travel to the northeast territories of India and wonder if there are any problems/restrictions for tourists. Find a website that is a good source of information about such problems/restrictions. *(T2)*

Figure 4: Tasks used in TREC 2002 interactive track experiments

4. Results and Analysis

As mentioned before, all users actions were logged. Most of the data analysed in this section came from these logs generated by the system during the interaction with the users. All statistical tests of significance are at $p \leq 0.05$, unless otherwise stated. *M* is used in this section to denote the mean.

4.1 Task Completion

As part of the TREC post-task questionnaire, users were asked to state whether they felt they had successfully completed the task just attempted. We believe that ultimately, it is the user's decision to state whether he completed a particular task or not. Indeed, this reflects real-life situations, where the purpose of any system is ultimately to satisfy the user. Roughly speaking, our assumption is that if a user has stopped the task within the 10 minutes allocated time, and said it has been successfully completed, it means the user is satisfied and the task is marked as completed. Table 1 shows the total number of failures for each system (out of 64).

Table 1: Levels of task failure on each system

	Baseline	HuddleSearch
Total number of failures	15	9
Average number of failures	1.875	1.125

Table 1 shows that the number of incomplete tasks is *clearly* reduced by the use of the experimental system HuddleSearch. This shows that the clustering and summarisation features aid the users in their interaction with the system. However, paired *T*-tests revealed that the difference between the two systems was not significant (T_{14} = 1.43, p = .195).

4.2 Task Times

The times taken to complete tasks on both systems were automatically measured from the user logs. Table 2 provides an overview of the performance of both experimented systems. A 60 second penalty is added when the task is incomplete. The systems calibration times are taken into account in Table 2. Indeed, while the average number of submitted query per task on each system is almost the same[5], HuddleSearch is on average slower ($M_{\text{difference}}$ = 9.7 seconds) than Panoptic in returning documents[6].

Table 2: Average time per task (seconds)

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8
Baseline	589.5	645	376.13	406.88	417.88	416.63	489.13	218.5
HuddleSearch	544.8	394.93	294.3	309.05	271.8	463.05	461.18	225.55

Compared to the Panoptic search engine, used as a baseline in our experimental design, the times taken to complete search tasks using HuddleSearch have significantly fallen (see Tables 2 and 3), especially for the first 5 tasks. On average, 74.4 seconds are saved per task using the HuddleSearch system. This difference is more marked for some tasks than others.

Table 3: Average task completion time per system (seconds)

	Baseline	HuddleSearch
Average task completion time	444.9531	370.5813

Two-way repeated measures ANOVA was run to test for a link between system, task and the associated time for each type of task. The results of this test showed that there was a significant difference between systems ($F_{7,112}$ = 7.16, p = .001), and a significant difference between tasks ($F_{7,112}$ = 9.34, p = .000). We found firstly that hierarchical clustering and summarisation visualisation techniques significantly help the users to locate quickly the relevant documents and secondly, that not all tasks were of equal difficulty. If we assume that task completion time is a reasonable indicator of task difficulty, then Task 1 (GR1 in Figure 4) was significantly more difficult than any of the other seven tasks, across both systems.

4.3 User Satisfaction

Overall, 13 out of 16 users preferred HuddleSearch to the baseline. Furthermore, as part of the post-task questionnaire, users were asked whether they were satisfied with the search results for each task. Table 4 shows that users do feel more satisfied by the results provided by our hierarchical clustering system ($M_{\text{huddlesearch}}$ = 4.468 vs. M_{baseline} = 4.219).

[5] About 3 queries per task on each system.

[6] Distributed systems technologies are currently investigated to cut down the answering time of HuddleSearch, which is essentially due to the multiple documents summariser component.

Table 4: Average user satisfaction with results for each task (Scale 1 to 7, higher = better)

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8
Baseline	3.875	1.5	5	4	4.625	4.625	4.25	5.875
HuddleSearch	2.75	4.5	4.5	6	5.125	3.625	3.625	5.625

However, these results are not significant using two-way, repeated measures ANOVA testing the effects of system ($F_{7,112}$ = .432, p = .764) and task ($F_{7,112}$ = .453, p = .717) on user satisfaction. The degree of user satisfaction is not significantly different between systems.

Given the very encouraging results we obtained for the tasks completion. the average completion times, and user satisfaction, we wanted to investigate what could improve the overall effectiveness of HuddleSearch. We propose some possible explanations as to why the system was not optimally efficient.

Firstly, Panoptic, the baseline search engine returns a document result set consisting of documents fully matching the query, followed by a set of results only partially relevant to the query (results are presented in tiers, in Panoptic terminology). When a user makes a request for precise information and asks for 300 pages to be returned, Panoptic might simply return 3 fully relevant documents, which are then occluded by a further 297 only partially relevant and not fully satisfying the query. In this way, many of the base clusters created may be too generic, and relevant to only a subset of the query terms, thus not aiding the user in his search.

A cluster is fundamentally a common group of terms; this has proved to conflict with Panoptic, which often returns mirrors of the same document several times. At this moment in time, HuddleSearch simply examines the URL to determine replication; consequently 'false' significant clusters are created when content is repeated, this will result in incorrectly weighted common phrases and will hide naturally relevant clusters which are pushed further down the list.

In order for a cluster summary to be created each page within a cluster is visited and its content is retrieved to generate a page summary. Some experiments were executed at peak times, where network traffic was high; hence many documents failed to be summarised within an allocated time. As a result, when documents failed to be summarised, the higher-level cluster summaries were inadequate and occasionally users were faced with 'no summary could be created', providing little or no benefit to the user. However, informal feedback from users during non-peak time evaluations suggests that summaries are indeed helpful.

In addition to viewing a cluster summary, our visualisation tool provides the facility of previewing a cluster by looking at its top documents titles, which we believe to be good indicators for judging initial relevance. However, the .GOV collection made this feature extremely temperamental, as in many cases a document's title is simply an alphanumeric document ID, thus providing no hint of content.

The above perhaps explain why the user satisfaction with HuddleSearch. though superior to the Panoptic engine, was not optimal. Moreover, the assessment of the answers provided by the users during the experiments will provide more evidence regarding the effectiveness of HuddleSearch. However, we do believe that most of the issues mentioned above could be addressed efficiently. Hence, we suggest that there is much scope to improve HuddleSearch. Overall, results show that hierarchical clustering and summarisation visualisation tools do aid the users in their interaction with the search engine in the Web context. Uncompleted tasks as well as average times to accomplish them were definitely reduced by the use of HuddleSearch.

Acknowledgements

We would like to thank the Research Committee of the Department of Computing Science for financially supporting this work. We also like to thank those who participated in the above experiments: their comments. feedback and enthusiasm were greatly appreciated.

References

[1] White. R.W., Jose, J.M. and Ruthven, I. 'Comparing Implicit and Explicit Feedback Techniques for Web Retrieval: TREC-10 interactive track report'. Proceedings of the Text REtrieval Conference (TREC 2001). Gaithersburg, Maryland, USA. November 2001

[2] Zamir, O. and Etzioni. O. 'Grouper: a dynamic clustering interface to Web search results'. In Proceedings of the Eighth International World Wide Web Conference. Toronto, Canada. May 1999.

University of Glasgow at the Web track of TREC 2002

Vassilis Plachouras, Iadh Ounis, Gianni Amati*, and C.J. Van Rijsbergen

Department of Computing Science
University of Glasgow
Glasgow G12 8QQ
{vassilis, ounis, gianni, keith}@dcs.gla.ac.uk

Abstract

The aim of our participation in the topic distillation and the named page finding tasks of the Web track is the evaluation of a well-founded modular probabilistic framework for Web Information Retrieval, which integrates content and link analyses. The link analysis component of the framework employs a new probabilistic approach, called the Absorbing Model, for calculating a measure of popularity for documents induced from the Web graph.

1 Introduction

Both topic distillation and named page finding tasks require a high precision in the ranking, since there are few relevant documents for the topic distillation task, which reduce to one for the named page finding task. Results of TREC last year [5] have shown that the link structure upon the Web hardly combines with the content analysis, at least for the topic relevance task. Similarly, the length of the URL and other heuristics have shown to be more effective than the explicit use of the link structure for the homepage finding task. In general, it has been found that pure link structure analysis does not enhance retrieval effectiveness.

Our participation in the Web track focuses on a novel way to integrate content with link analysis. We propose a method for combining content and links in a sound and non-parametric way, as well as a dynamic spreading activation mechanism to be applied on top of this methodology. We evaluate our proposals by extending the framework used in TREC10 [1] in a modular way. The refined content analysis module is integrated with a new link analysis module. Moreover, the system is extended with a new query expansion module, which can be enabled, or disabled during the retrieval process.

The preliminary results reported in this paper show that link analysis may benefit the ranking, enhancing the overall retrieval effectiveness. However, its success strongly depends on the basic probabilistic link model used, as well as on the methodology employed to build the link structure graph. Our approach differs from that used in PageRank [4] and it can be seen as a dynamic modification of the link structure according to the content and thus, indirectly, to the query. Moreover, results show that different strategies are required for optimal topic distillation and named page finding effectiveness.

The rest of the paper is organised as follows. In Section 2, the proposed approaches and their rationale are described. Section 3 contains a detailed description of the runs submitted, and in Section 4 we present a preliminary analysis of the results.

*Also affiliated to Fondazione Ugo Bordoni, Rome.

2 Integration of content and link analyses

We introduce a new probabilistic model, which we call the Absorbing Model, for analysing the link structure of Web documents [2]. This method provides us with a measure of authority for documents, based on the probability distribution of accessing the states of a Markov chain, which is induced from the Web graph. This probability distribution corresponds to the principal eigenvector of the adjacency matrix of the Web graph, and it is computed by an iterative but necessary *converging* process. We apply this method either statically or dynamically.

2.1 A parameter-free link model: the Absorbing model

PageRank [4] connects any couple of documents in the collection through a virtual link, with a very small transition probability, which is a parameter for the algorithm. In the terminology of Markov chains, all documents on the Web become ergodic states, which are treated as nodes belonging to a unique large cluster. This cluster guarantees that all documents receive a probability different from zero (the authoritative score), which is the probability that the node is reached by any possible random walk through the nodes of the collection. Our Markov chain model offers a completely different solution. We transform the original graph to make all states ergodic and have many different clusters instead of one. The transition probabilities are then computed accordingly. In this transformation process, we do not need the use of parameters to obtain the final authoritative scores.

2.2 Static application of the Absorbing model

For the static application of the Absorbing Model (SAM), we first calculate an absolute authority score for each document in the collection during indexing. After retrieval is performed, this authority score is combined with the content based score in a parametric way. This approach is similar to the PageRank philosophy.

2.3 Dynamic application of the absorbing model

The dynamic application of the Absorbing model, which is called Dynamic Absorbing Model (DAM), seems to be promising. It is applied only on the top retrieved documents and aims to provide a dynamic authority measure. The retrieved documents are returned by the application of a classical content analysis model. Using a high quality content retrieval module, we assume that most of the authoritative documents will be among the top ranked documents, and thus the application of link analysis may change, but not drastically, the ranking of documents. Following this idea, the DAM is applied on the set B of the top $|B|$ ranked documents. The priors are initialised using the scores of the content analysis. Among the top retrieved documents, we select the subset $A \subset B$, with $0 \leq |A| < |B|$, with the highest content scores. We suppose that the documents in this set A should be more authoritative than the remaining ones in $B - A$. Therefore, we modify the subgraph of the subset A of the top ranked documents of B by removing their outgoing links. In this way, the documents in the set $B - A$ that might be relevant are boosted according to the link structure of $|B|$, while documents in A do not loose their authority score but they may inherit some more from $B - A$. The application of DAM does not involve any training, which makes it suitable for large collections of documents, such as the Web.

2.4 Spreading Activation

In addition to the Absorbing Model, we propose a dynamic spreading activation mechanism for finding the best entry points for topics on the Web.

Spreading activation is a well known mechanism used in hypertext and web retrieval systems [8, 6], where a fraction of the Retrieval Status Value (*RSV*) of each document propagates to the documents linked by it, assuming that documents linked by other relevant documents, are possibly relevant as well. We adapt this mechanism in order to fit our need for identifying the best entry points for a topic. In our method, which we call Static Spreading Activation (SSA), the propagation of *RSV*s is performed only when the documents linked belong to the same domain and the source document of the link is deeper in the document tree than the destination document.

2.5 Query-biased Spreading Activation

The above spreading activation mechanism is refined by allowing the fraction of the RSV that propagates to vary, depending on a measure of the query specificity. This measure, which we call *query scope*, is a hybrid probabilistic measure that depends on both collection statistics and the external conceptual structure provided by WordNet [7]. It is based on the assumption that a generic query consists of terms that correspond to generic concepts in a conceptual hierarchy, and also that occur frequently in the collection. In the dynamic version of the spreading activation mechanism, which we call Dynamic Spreading Activation (DSA), the query scope is used to adjust the effect of the *RSV* propagation, according to the assumption that generic queries may benefit more from the link structure analysis, and therefore from the propagation of a larger fraction of the *RSV*s between linked documents. During indexing, each term appearing in the collection is mapped to one or more concepts in WordNet and a probabilistic measure of the specificity of the term is calculated. During retrieval, the sum of the query term specificity measures, normalised by the length of the query, represents a probabilistic measure for the specificity of the query, which is used to refine the application of the spreading activation mechanism.

3 Description of experiments

3.1 Indexing

For indexing the collection, a standard stop word list is used and Porter's stemming algorithm is applied. Moreover, in all runs, except for uog05tad and uog06c, the documents are modified by doubling the occurrences of terms in titles and by adding to each document the anchor text of its incoming links. For the run uog06c, the documents are not modified, while for the run uog05tad, the inverted file is augmented with additional information on whether a term appears in the anchor text of a document or in its body. For all runs, the weighting formulas used for calculating the *RSV*s are a *variant* of $I(n_e)B2$, which we call $I(n_e)C2$, except for runs uog04cta2dqh and uog09cta2, where the formula $I(n_e)B2$ is used [3].

3.2 Topic Distillation

The static Absorbing Model (SAM) is applied only in the run uog01ctaialh for the topic distillation task. In more details, the probability distribution of accessing the states of a modified Markov chain obtained from the graph of the collection is calculated during indexing. We take into account only links between documents that belong to different domains. During retrieval, after forming the elite set, consisting of the top 1000 retrieved documents, we calculate a new RSV' using a linear combination of the RSV initially computed and the Absorbing Model's authority score $auth_{AM}$:

$$RSV' = a_{content} * RSV + a_{link} * auth_{AM} \qquad (1)$$

Official run	Prec. at 10	Prec. at 20	Prec. at 30	Average Prec.	Features
`uog01ctaialh`	0.1306	0.1255	0.1218	0.1072	body-anchor-title , $I(n_e)C2$ SAM, SSA
`uog02ctadh`	0.1143	0.1184	0.1116	0.0979	body-anchor-title, $I(n_e)C2$ DAM ($\|B\| = 50, \|A\| = 10$), SSA
`uog03ctadqh`	0.1939	0.1612	0.1476	0.1582	body-anchor-title; $I(n_e)C2$ DAM ($\|B\| = 50, \|A\| = 10$), DSA
`uog04cta2dqh`	0.2082	0.1704	0.1469	0.1743	body-anchor-title , $I(n_e)B2$ DAM ($\|B\| = 50, \|A\| = 10$), DSA
`uog05tad`	0.2224	0.1765	0.1565	0.1540	body-anchor-title, $I(n_e)C2$ DAM ($\|B\| = 50, \|A\| = 10$)

Table 1: Topic distillation official results

The parameters $a_{content}$ and a_{link} were experimentally set to 1 and 0.1 respectively.

The DAM is applied in runs `uog02ctadh`, `uog03ctadqh`, `uog04cta2dqh` and `uog05tad` for the topic distillation task. After the elite set of documents is ranked according to the matching function used, the $RSVs$ calculated are used as prior probabilities for the initialisation of the DAM. This dynamic link analysis is applied on the set B of the top ranked documents, ignoring the outlinks from the documents of the set A, where $A \subset B$. The values used in the official results for the sizes of sets B and A are respectively 50 and 10.

The spreading activation mechanism is applied for the topic distillation task, either statically for runs `uog01ctaialh` and `uog02ctadh`, or in a dynamic mode for runs `uog03ctadqh` and `uog04cta2dqh`, as a filter for re-ranking the results. The static version (SSA) consists of forming for each document d in the elite set, the set S of documents that are linked by it and that are placed deeper in the hierarchy of documents within the same site. The final RSV' for document d is then calculated by using the following formula:

$$RSV'_d = RSV_d + \beta * \sum_{s \in S} RSV_s \tag{2}$$

The parameter β was experimentally set to 0.1 .

The dynamic version (DSA) of the spreading activation mechanism replaces the parameter β with the query scope, that is a measure of the specificity of the query. The Equation 2 is then modified as follows:

$$RSV'_d = RSV_d + query_{scope} * \sum_{s \in S} RSV_s \tag{3}$$

As mentioned above in Section 3.1, for the run `uog05tad`, we use a different approach. Having augmented the inverted file with information on whether the terms belong to the body of a document or to its anchor text, before the DAM is applied, we remove from the results those documents for which no query terms occur in the associated anchor text. The obtained official results for the topic distillation task are given in Table 1.

3.3 Named page finding

For the named page finding task, proximity search is used only for the run `uog08ctap`. After retrieval is performed and the elite set is formed, proximity search is applied and the original RSV of documents in which the query occurs as a phrase is multiplied by a parameter γ, as shown in the following equation:

$$RSV' = \gamma * RSV \tag{4}$$

Official run	Average Reciprocal Precision	Named pages in top 10	Named pages not found	Features
uog06c	0.552	107 (71.3%)	23 (15.3%)	body only, $I(n_e)C2$
uog07cta	0.654	128 (85.3%)	14 (9.3%)	body-anchor-title, $I(n_e)C2$
uog08ctap	0.516	114 (76.0%)	18 (12.0%)	body-anchor-title, $I(n_e)C2$ Proximity search
uog09cta2	0.643	127 (84.7%)	12 (8.0%)	body-anchor-title, $I(n_e)B2$
uog10ctad	0.651	128 (85.3%)	14 (9.3%)	body-anchor-title, $I(n_e)C2$ DAM ($\|B\| = 10, A = \|5\|$)

Table 2: Named page finding official results

Unofficial run	Prec. at 10	Prec. at 20	Prec. at 30	Average Prec.	Features
unof01cta	0.2082	0.1714	0.1537	0.1685	body-anchor-title, $I(n_e)C2$
unof02c	0.2122	0.1806	0.1619	0.1668	body only, $I(n_e)C2$
unof03c	0.2694	0.1929	0.1680	0.2041	body only, $PL2$
unof04cd	0.2776	0.1969	0.1687	0.2047	body only, $PL2$ DAM ($\|B\| = 50, \|A\| = 20$)
unof05cdpr	0.1939	0.1612	0.1463	0.1335	body only, $PL2$ Dynamic PageRank ($\|B\| = 50, \|A\| = 20$)
unof06cqe	0.2388	0.1888	0.1653	0.2021	body only, $PL2$ Query Expansion

Table 3: Topic distillation unofficial results

For the run uog08ctap the parameter γ was set to 1.3.

For the run uog10ctad, the DAM is applied as described in Section 2.3. The sizes of the sets B and A were experimentally chosen to be 10 and 5. They are smaller than the corresponding values chosen for the topic distillation task, reflecting the fact that there is only one, or two at most named pages for each query.

The rest three runs, namely uog06c, uog07cta and uog09cta2 are variations of body only, or body and anchor indexing retrieval, using different retrieval methods. Run uog06c uses body only indexing, while runs uog07cta and uog09cta2 use body and anchor indexing. Moreover, runs uog06c and uog07cta use the retrieval method $I(n_e)C2$, while for the run uog09cta2 the method $I(n_e)B2$ was applied. The obtained official results for the named page finding task are given in Table 2.

4 Analysis of results

This year, both tasks require a high early precision. The number of relevant documents for the topic distillation task is smaller than the number of relevant documents found for the topic relevance task in TREC 10. The named page finding task differs from the homepage finding task, because the named pages are not necessarily homepages.

For the sake of completeness, once we received the evaluation data from TREC, we ran several new unofficial experiments (see Tables 3 and 4).

Both official and unofficial results obtained from the conducted experiments, show that the content analysis is still the most important/efficient retrieval component. For example, in the topic distillation task, our content-only baseline, unofficial run unof03c, as shown in Table 3, performs better than all our official runs (0.2694 of prec. @10 obtained by run unof03c w.r.t. 0.2224 obtained by

Unofficial run	Average Reciprocal Precision	Named pages in top 10	Named pages not found	Features
unof07ctad	0.555	107 (71.3%)	22 (14.67%)	body only, $I(n_e)C2$ DAM ($|B| = 10, |A| = 5$)
unof08cqe	0.414	93 (62.0%)	38 (25.3%)	body only, $I(n_e)C2$ Query expansion
unof09cta	0.614	124 (82.67%)	11(7.33%)	body-anchor-title, $PL2$

Table 4: Named page finding unofficial results

our best run, uog5tad). Note here the application of the weighting scheme $PL2$, which we found to clearly outperform others schemes mentioned in [3].

To have a clear view of the importance of link analysis in topic distillation, we have tuned our link analysis model, the DAM, running several experiments with different values for the sets B and A. We have observed a slight improvement of precision at 10 documents and average precision with respect to the content only baseline using the DAM (0.2776 of prec. @10 obtained by run unof04cd w.r.t. 0.2694 obtained by the pure $PL2$ content retrieval baseline, run unof03c, as shown in Table 3). This result was not achieved with the official runs.

Amongst the official runs, uog05tad was the best. Performance difference between the official run uog05tad and the unofficial run unof04cd was due to the fact that in the official run, the use of the anchor and the title text was detrimental, the size of A was too small for a precision @10 and the content retrieval model was different.

We have also compared the DAM to PageRank under the same experimental setting for the topic distillation task. PageRank is applied on the set B of the top $|B|$ documents and the outlinks of the top $|A|$ documents are ignored, where $0 \leq |A| < |B|$. Results show that DAM significantly outperforms PageRank for different values of $|B|$ and $|A|$ (e.g. run unof05cdpr w.r.t. run unof04cd in Table 3). Moreover, PageRank seems to be detrimental for topic distillation (run unof05cdpr w.r.t. run unof02c in Table 3).

We also achieve a slight improvement over the body-only indexing retrieval baseline for the named page finding task (run unof07ctad in Table 4 w.r.t. run uog06c in Table 2) by using our link analysis model, DAM. Although the improvement is marginal, and the sizes of the sets B and A on which the link analysis was applied are smaller than the corresponding sizes for the topic distillation task, this is an indication that our dynamic link analysis model may be applied for both tasks.

An interesting issue that arises from the conducted experiments, concerns the use of anchor text during retrieval. While for the named page finding task, employing anchor text significantly improves precision (run uog07cta w.r.t. run uog06c in Table 2), for the topic distillation task precision decreases (run unof02c w.r.t. run unof01cta in Table 3).

Also, the query-biased spreading activation mechanism proposed seems to significantly improve results over the statically applied spreading activation (run uog03ctadqh w.r.t. run uog02ctadh in Table 1), although the effectiveness is lower than that of our baseline. A possible reason for this is that the spreading activation mechanism was applied on the set of the 1000 top ranked documents, where not all the links contained are useful and most of the documents are not relevant. Therefore, additional refinements are needed, as well as an investigation on the size of the set of documents for which the spreading activation mechanism will prove to be effective. Furthermore, we have noted that query expansion is detrimental for both tasks (run unof06cqe in Table 3 for topic distillation and run unof08cqe in Table 4 for named page finding).

Moreover, the conducted experiments show that the two tasks of the Web track are intrinsically different. Thus, different strategies are needed for each one, since what works best for one task is not necessary optimal for the other. First, the best results in the two tasks were obtained by using different weighting schemes, i.e. $PL2$ works better for topic distillation (unofficial run `unof03c` w.r.t. unofficial run `unof02c` in Table 3), while $I(n_e)C2$ performs the best in named page finding (official run `uog07cta` w.r.t. unofficial run `unof09cta`). Second, we have proved that while using anchor text improves precision for the named page finding task, it decreases performance in the topic distillation task.

To conclude, for the topic distillation task, both body-only indexing and link analysis without anchors work well, whilst for the named page finding task body and anchor indexing also display promising results. Furthermore, our results show that the potential application and usefulness of the link analysis still has to be explored, and we believe that performance improvement is feasible through refinement and better integration of the content and link analyses. As far as we know, even though content analysis is still a major component for effective retrieval, it is the first time that the results benefit from the application of pure link analysis for both tasks.

Acknowledgments

This work is funded by a UK Engineering and Physical Sciences Research Council (EPSRC) grant, number GR/R90543/01.

References

[1] G. Amati, C. Carpineto, and G. Romano. FUB at TREC 10 web track: a probabilistic framework for topic relevance term weighting. In E.M. Voorhees and D.K. Harman, editors, *Proceedings of the 10th Text Retrieval Conference TREC 2001*, pages 182–191, Gaithersburg, MD, 2002. NIST Special Pubblication 500-250.

[2] G. Amati and I. Ounis. The absorbing link model for the web. *Manuscript*, 2002.

[3] G. Amati and C. J. Van Rijsbergen. Probabilistic models of information retrieval based on measuring divergence from randomness. *ACM Transactions on Information Systems*, 40(4):1–33, 2002.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[5] D. Hawking and N. Craswell. Overview of the TREC-2001 Web Track, NIST Special Publication 500-250: The Tenth Text REtrieval Conference (TREC 2001), 2001.

[6] R. Jin and S. Dumais. Probabilistic combination of content and links. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 402–403. ACM Press, 2001.

[7] V. Plachouras and I. Ounis. Query-Biased Combination of Evidence on the Web. Workshop on Mathematical/Formal Methods in Information Retrieval, ACM SIGIR Conference, 2002.

[8] J. Savoy and J. Picard. Retrieval effectiveness on the web. *Information Processing & Management*, 37(4):543–569, 2001.

Incremental Retrieval
# of documents relevant to a topic

Caroline Lyon, Bob Dickerson, James Malcolm
University of Hertfordshire, UK
c.m.lyon@herts.ac.uk

Introduction

As new participants to TREC, on the Filtering Track, we have started by first investigating two methods of producing document profiles. We begin by looking for "obvious" profiles that detect closely related documents. This year we have started by looking for:

- lexically similar cases
- semantically similar cases based on a simple combination of keywords.

Characteristics of the Reuter's data

Before addressing specific tasks we investigated the Reuter's data. It was expected in this domain that there would be some similar text in different documents: the extent is quite significant. We used the Ferret software, designed to ferret out similar passages of text in large document collections, which we have recently developed [2].

An experiment was carried out to compare each document with about 1000 others, taken in date order. We went through the test corpus (723141 documents) and for every set of 1000 documents compared each with each (that is 499500 comparisons for each set). Of course if file A is similar to file B and to file C, then it is quite likely that File B is similar to file C.

We found 48,918 with identical text. Some of the files were very short, for instance regular industrial reports might have no more than 10 content words in the text. Omitting files with 10 or less content words, 6,616 had identical text.

The analysis also showed that in a further large number of file pairs texts were "very close" – this and other terms will be explained below. 287,391 pairs fell into this category. Without those files containing 10 or less content words in their texts, 24,017 were very close.

There are 718, 443 pairs with "significant matching passages". Of those with more than 10 content words in the text 228,130 fall into this category.

Method of determining similarity

The method used is as follows. First each document is pre-processed so that only the id number, the headline, and the text are kept, while tags are omitted. Stop words are filtered out. There are 440 stop words, and the list includes entries which, though not function words, have little semantic content.

Then each document is converted into a set of word triples, composed of every sequential triple. Thus, the sentence:

Given a topic description and some example relevant documents build a filtering profile.

would be converted into the set:

given a topic   a topic description   topic description and   etc.

or, after taking out stop words:

given topic description   topic description example   description example relevant etc.

Then each pair of documents is compared for matching word triples. This raw score is converted into the metric "resemblance", based on set-theoretic principles. Informally, resemblance is the number of matches between two sets, scaled by joint set size. It is also known as the Jaccard coefficient. Let S(A) and S(B) be the set of trigrams from documents A and B respectively. Let R(A,B) be the resemblance between A and B

$$R = \frac{|S(A) \cap S(B)|}{|S(A) \cup S(B)|}$$

For the preliminary investigations into the Reuter's data, documents are identical if, after pre-processing, R =1.0. The category "very close" takes $1.0 > R \geq 0.8$, while "significant matching passages" takes $0.8 > R \geq 0.4$. These are arbitrary boundaries.

As an indication of the scale of similarity, it is worth considering measures used in another field. The Ferret was originally developed for detecting plagiarism in students' work. At a level of $R > 0.04$ (a degree of magnitude smaller than that used here) matching passages were typically found, possibly quite short.

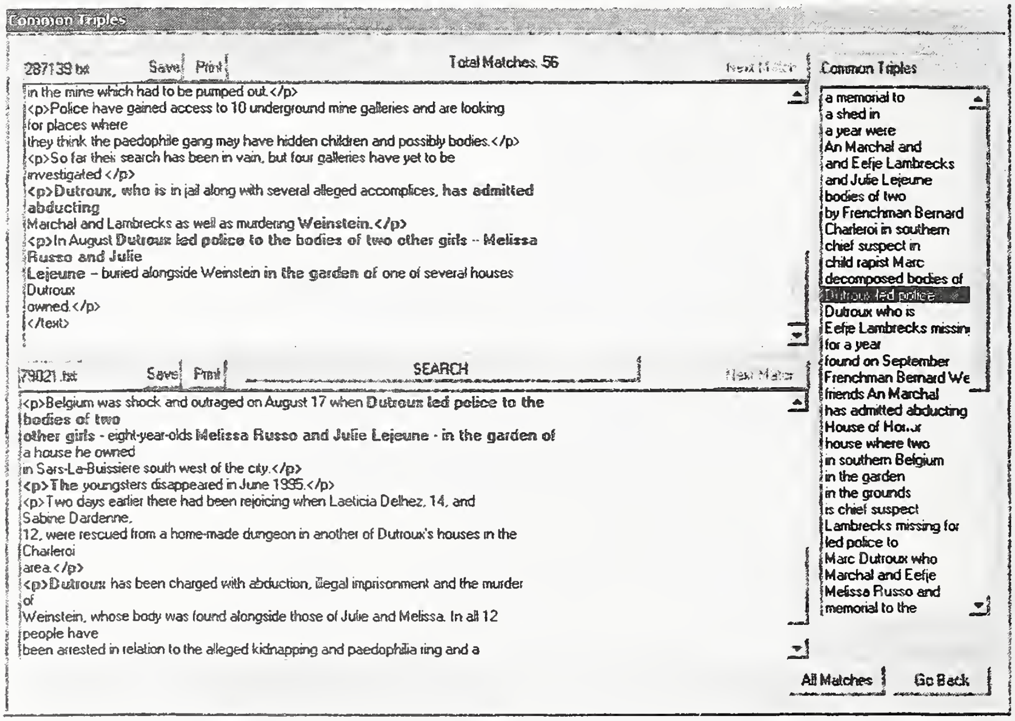
Time taken to process each set of 1000 files was about 1 minute, about 11 hours for the full test set, on a Pentium III processor, with 700MHz, 512 MB RAM. However there is considerable scope for increasing the efficiency of this implementation.

Theoretical background

The dominant approach in statistical pattern analysis is based on the well known method of abstracting significant features and lining them up in a feature vector for further processing. However, there are relationships between the number of elements of the feature vector, the amount of training data available and the level of generalization achieved. In text processing a very large number of words have to be processed, even after filtering through a stop word list. The amount of training data will typically not be enough to ensure a satisfactory level of probably approximately correct outcomes. For further details see [1, 3]. Therefore, a set theoretic approach may be appropriate in word based text processing, as described in [2].

Routing filtering with lexical profiles

The method described above was then applied to give a preliminary analysis of topics in the filtering task. For this we just took the three sample documents given for the adaptive filtering task, and did not refer to the topic description. The three sample documents are stripped of xml tags, edited by filtering through the stop word list and concatenated. This text is then compared to all the documents in the test data (similarly detagged and filtered through the stop word list). For Topic 102 a pairing producing 16 matches, resemblance 0.05, is displayed, Figure 1. The number of matching word triples shown in the display is much greater than that produced by the match detection software, since for display we go back to the original documents which include stop words and xml tags.



```
in the mine which had to be pumped out.</p>
<p>Police have gained access to 10 underground mine galleries and are looking
for places where
they think the paedophile gang may have hidden children and possibly bodies.</p>
<p>So far their search has been in vain, but four galleries have yet to be
investigated </p>
<p>Dutroux, who is in jail along with several alleged accomplices, has admitted
abducting
Marchal and Lambrecks as well as murdering Weinstein.</p>
<p>In August Dutroux led police to the bodies of two other girls -- Melissa
Russo and Julie
Lejeune -- buried alongside Weinstein in the garden of one of several houses
Dutroux
owned.</p>
</text>
```

```
<p>Belgium was shock and outraged on August 17 when Dutroux led police to the
bodies of two
other girls - eight-year-olds Melissa Russo and Julie Lejeune - in the garden of
a house he owned
in Sars-La-Buissiere south west of the city.</p>
<p>The youngsters disappeared in June 1995.</p>
<p>Two days earlier there had been rejoicing when Laeticia Delhez, 14, and
Sabine Dardenne,
12, were rescued from a home-made dungeon in another of Dutroux's houses in the
Charleroi
area.</p>
<p>Dutroux has been charged with abduction, illegal imprisonment and the murder
of
Weinstein, whose body was found alongside those of Julie and Melissa. In all 12
people have
been arrested in relation to the alleged kidnapping and paedophilia ring and a
```

Common Triples

a memorial to
a shed in
a year were
An Marchal and
and Eefje Lambrecks
and Julie Lejeune
bodies of two
by Frenchman Bernard
Charleroi in southern
chief suspect in
child rapist Marc
decomposed bodies of
Dutroux led police
Dutroux who is
Eefje Lambrecks missin
for a year
found on September
Frenchman Bernard We
friends An Marchal
has admitted abducting
House of Ho..r
house where two
in southern Belgium
in the garden
in the grounds
is chief suspect
Lambrecks missing for
led police to
Marc Dutroux who
Marchal and Eefje
Melissa Russo and
memorial to the

Figure 1: From Topic 102, display of sample text 79021 and relevant document 287139

In Figure 1 the lower of the two files, id 79021, is one of the 3 example documents for Topic 102. The upper file, id 287139, has short passages of matching text. It seems that the original story was picked up again some time later.

This illustration shows how short passages of matching text can be detected. Lexically similar text is often semantically similar too. However, this is not always the case, as when the processor picks up commonly occurring comments such as "Reuters has not verified these reports and cannot vouch for their accuracy".

The type of lexical similarity described above indicates semantic similarity. However, the opposite is not true. If two people write on the same topic independently the resulting articles will not be lexically similar in this way, as previous experiments have shown. When texts are lexically similar it indicates that there has been some element of cutting and pasting.

Routing filtering with simple keyword profiles

The concept behind this method is to have several sets of keywords, and for a document to be considered relevant it must have at least one member in each set. The keywords have been selected manually at this point, from the topic description and three sample documents for the adaptive filtering task. The rest of the training data was used for primary evaluation of this approach. Topics R101 and R125 were entered on this track. For initial work there were 3 sets of keywords. It was essential to have a member in sets key1 and key2. Key3 was a set of supporting keywords whose frequency of occurrence determined the ranking. As an example, the keywords for Topic R101 on industrial espionage were as follows:

key1	key2
espionage spy spying	business commercial economic industrial technical

Figure 2 : Essential keywords

Using this method cuts down on possible combinatorial explosion of combinations of terms "industrial espionage", "commercial espionage", "industrial spying" etc. On inspection later, it seemed that key1 might have included "secrets" and key2 "company". This would have caught some documents that slipped through the net, but might have produced false positives too.

key3	
charges	police
confidential	prosecution
court	prosecutor
courts	prosecutors
covert	secret
intelligence	secrets
investigation	surveillance

Figure 3: Non-essential keywords used for ranking

Results

Using this method on topic R101 produced a score of 0.428 compared to median 0.469 and maximum0.902. On R125 it produced a result of 0.062, compared to a median of 0.327 and maximum of 0.565.

In both case the number of relevant documents was well below the specified number. For R101 477 were found, for R125 260 were found. However, limited random sampling indicated that no false positives were found.

Discrepancies in the data

In some cases the topic description and the training documents were not consistent. For example, Topic R110 was entitled "Terrorism Middle East tourism", and the narrative said relevant documents should correlate terrorism with tourism. However, "terrorism" and associated terms were not mentioned in the 3 training documents for adaptive filtering (42439. 82926, 85147). Topic R125 was entitled "Scottish Independence" but there was no mention of Scotland in any form in some documents judged relevant (27974, 48375, 68664). In Topic 134 the narrative of the topic description said that documents were relevant only if statistics were included. There were no statistics in one of the three training documents (73372).

Conclusion

The first method employed detected little of that lexical similarity between training and testing documents, which is indicative of re-using text. However, our investigation of general characteristics of the data showed that there is much re-use of text on close dates.

Taking a sideways glance at the Novelty Track, this method could be useful to sort out similar versions of a story from ones with new information. Whether the new information is strictly relevant would be another matter. For instance, reports on ABA banking policy (100017.100398) had similarities (resemblance 0.55). The second had additional information, on the speakers' clothes, which might not be considered relevant.

The second method employed, using combinations of keywords, is a useful way of a detecting a core of relevant documents. This could possibly be automated using thesauri and/or Wordnet.

If the Filtering track is reinstated we plan to move on to the more interesting hard-to-detect cases, and to integrate different profiles as in co-training.

References

1. A K Jain, R P W Duin and J Mao. Statistical Pattern Recognition: A review. *IEEE Trans. on Pattern Analysis and Machine Intelligence, 22. 1. 2000.*

2. Caroline Lyon, James Malcolm, and Bob Dickerson, Detecting short passages of similar text in large document collections, *Proc. of Conference on Empirical Methods in Natural Language Processing.* 2001.

3. C Lyon and R Frank. Using single layer networks for discrete sequential data: an example from Natural Language Processing. *Neural Computing Applications 5 (4) 1997*

UIC at TREC-2002: Web Track

Shuang Liu, Clement Yu, *Wensheng Wu

Database and Information System Lab
Computer Science Department, University of Illinois at Chicago
{sliu, yu}@cs.uic.edu

*Computer Science Department
University of Illinois at Urban and Champion
wwu2@uiuc.edu

Abstract

This is the first year that members of the Database and Information System Lab (DBIS) at University of Illinois at Chicago (UIC) participate in TREC. We participate in two tasks for the Web track: topic distillation and named page finding. Linkage information among documents as well as content information about documents is used in some of our submitted runs.

We utilize the Okapi weighting scheme with some modification for documents and passages retrieval; the proximity of query terms in documents is also utilized for document ranking.

The PageRank of a document is combined with the similarity of the document with the query to obtain an overall ranking of documents. A local linkage and URL analysis algorithm is employed for topic distillation. In the named page finding task, we combine the surrogate similarity with the document similarity in one run.

0. Introduction

In TREC-2002 experiments, we carry out the topic distillation and the named page finding tasks. A modified Okapi weighting scheme is implemented in our search engine. The modification is to replace the original parameter *K* which is the length of a passage by the parameter *K'* which is the norm of the passage. We also propose a proximity re-ranking method. In this method, documents covering more query terms in a certain window size than documents having fewer query terms within the same window size are ranked higher. The PageRank of a document is combined with the similarity of the document before or after proximity re-ranking to obtain an overall ranking of the document. In topic distillation, a document is assigned a value which is the sum of its similarity and a weighted sum of the similarities of its descendents within the same host. For each host, the document that gets the highest value among documents belonging to the same host is chosen for final ranking. In named page finding, the title and the anchor text are used to construct a surrogate index. We submit one run that combines the document similarity with the surrogate similarity.

In section 1 of this paper, we present the indexer, the weighting schemer, the proximity re-ranked feature and the linkage analysis. In the next part, we discuss the specific technique we used for the distillation task and the named page finding task. The paper is concluded in section 3.

1. System Description

1.1 System and Data Structure

In this section, an overview of our system is given.

System

(1) Document Parser

It is responsible for robust parsing of HTML/SGML documents. This includes HTML/SGML parsing, tokenization, lower case conversion and stemming.

(2) Indexer

The indexer reads collection files from disk, decompresses the documents, and parses them. For each document, the system assigns a document ID, stores the content words (word ID) and their positions within the document. For each appearance of the content word, its type (title, anchor or plain) is identified. The information is held in a forward index file, in descending order of document ID. The lexicon will be kept in a separate file. The lexicon file contains all the words and their corresponding IDs. The indexer also extracts linkage information between documents and maintains it by using two tables: one keeps track of, for each document, the set of documents it points to; the other keeps track of, for each document, the set of documents pointing to it.

(3) Inverse Indexer

The inverse indexer reads the forward index file and sorts it by word ID to generate an inverted index. An index of the inverted index is produced which contains, for each word, the word ID and its offset into the inverted index.

A document is partitioned into passages, each having at most 300 words including stop words.

Data Structure

(1) Forward Index

Documents in the forward index are in ascending order of document ID. One complete record is as follows:

Document ID, t

Word Id_1, tf_1, (position, type[, docID]), …., (position, type[, docID])

Word Id_2, tf_2, (position, type[, docID]), …., (position, type[, docID])

…

Word Id_t, tf_t, (position, type[, docID]), …., (position, type[, docID])

where t is the number of distinct terms in the document; tf_i is the term frequency within the document; for docID_{i1} in (position, type[, docID_{i1}]), if the word type is anchor, the word is logically associated with the document with docID_{i1}; otherwise the docID_{i1} is omitted.

(2) Inverted Index

Ordered by word ID, the word record is as follows:

Word ID, df, (docID, normWeight) ….. (docID, normWeight)

where df is the document frequency of the word; docID and normWeight are the document and the normalized term weight of the word in the document respectively.

There are exactly two forward index files and two inverted index files, one for documents and another for passages.

(3) Outgoing Link File and Back Link File

Ordered by document ID, the outgoing link record is as follows:

Document ID, t, docID_1, docID_2, …, docIDt

where t is the number of outgoing links; documents with IDs: docID_1, docID_2, … docID_t are pointed to by document ID.

The back link file has the same format, but the docIDs are associated with the back links of the given document.

(4) Document Index

Document index table, implemented as a fixed length ISAM file and ordered by document ID, contains, for each document, document ID, document name, its URL, its length, its norm, its

PageRank (see section 1.4) and entries to various files: forward index file, outgoing links file, and back links file.

Basic Search Engine

In the current system, once the search engine is fired up, several data structures are populated and resident in main memory. This includes (a) a hash table holding lexicon; (b) a set holding a list of stop words; and (c) a search map holding the index of the inverted index. The search map is implemented as sorted pairs of (word ID, offset), where the offset indicates the starting position where the inverted index for the word with word ID is located on the disk.

A memory resident lexicon and search map enables fast lookup of inverted file and is critical to the efficiency of our search engine.

All the processing is done on a Dell computer with 2.0G Pentium processor and 1GB RAM. This machine runs Linux.

1.2 Weighting Scheme

We adopt the Okapi weighting function in our system [RobertsonWalker00].

Basic Weight Function

$$\sum_{T\in Q} w^{(1)} \times \frac{(k_1+1)\times tf}{K+tf} \times \frac{(k_3+1)\times qtf}{k_3+qtf} \qquad (1)$$

$w^{(1)}$ is the Robertson/Spark Jones weight of T in Q [RobertsonSpark76], which is:

$$\log\frac{N-n+0.5}{n+0.5} \qquad (2)$$

where,

N is the number of items (documents/passages) in the collection

n is the number of documents/passages containing the term

tf is the frequency of occurrence of the term within a specific document/passage

qtf is the frequency of the term within the query

$K = k_1 \times ((1-b) + b \times \frac{dl}{avgdl})$, $k_1 = 1.2$ $b = 0.75$ $k_3 = 1000$

Documents retrieval uses the basic okapi function.

Modified Okapi Function for Passages Retrieval

Passages retrieval uses a modified okapi function where each passage has up to 300 words including stop words (the last passage of a document may have less than 300 words).

We modify Okapi function by replacing the original parameter *K* with *K'*, where *K'* is:

$$K' = k_1 \times ((1-b) + b \times \frac{Norm}{AvgNorm})$$

Here the Norm is the norm of the passage, the AvgNorm is the average norm of a passage in the collection. k_1, b, k_3 are 1.2, 0.75 and 1000 respectively. We use the Norm and AvgNorm instead of.the traditional length factor for the reason that passages differ more in norms than in lengths.

1.3 Proximity Feature

We take the proximity of query terms into consideration in retrieval. Specifically, if the largest number of query terms which can be found in a document within a certain window size (50 words) is m, then m is called the proximity factor. Supposed that the similarity of the document computed by the basic/modified Okapi function is sim, then an intermediate similarity of the document is (m, sim). Documents are arranged in descending order of (proximity factor, similarity) with proximity factor being the leading coefficient. That is, if document d_1 has intermediate similarity (p_1, sim_1), and document d_2 has intermediate similarity (p_2, sim_2), then document d1 will be ranked higher than document d_2 if $p_1 > p_2$, or $p_1 = p_2$ and $sim_1 > sim_2$.

After TREC competition, we do more experiments on proximity. We are investigating the effect of window size on retrieval efficiencies.

1.4 PageRank

We compute the PageRank of each document according to [BrinPage98] [PageBrinMotwani98] [Havelinwala99].

1.5 Retrieval

Documents/Passages Retrieval

Documents are retrieved in descending order of basic okapi similarity. Let the similarity between document d_i and query q be normalized to value between 0 and 1 and be denoted by sim(di, q).

The basic search engine also retrieves the passages in descending order of the modified okapi similarity. The similarity of a document containing a set of passages in the list is the maximum similarity of a passage within the document. Let sim(p_k, q) represent the similarity value between passage p_k and query q, p_{i1}, p_{i2}, …, p_{it} be passages belonging to the same document d_i. The similarity of document d_i between query q is normalized between 0 and 1 and is given by:

$$\text{sim}(d_i, q) = \max(\ \text{sim}(p_{i1}, q), \text{sim}(p_{i2}, q), \ldots, \text{sim}(p_{it}, q)\ ) \tag{3}$$

Ranking Documents by Proximity

After the similarities of documents are computed according to formula (3), they are re-ranked in descending order of (proximity factor, similarity) as described in section 1.3. The (proximity factor, similarity) is normalized to a value between 0 and 1, and this value is denoted by inter-sim(d, q).

Combine PageRank

We combine the PageRank value of each document with sim(d, q) or inter-sim(d, q) to aim at the final similarity [SilvaRibeiro-Neto00].

Combine with PageRank:

$$\text{final-sim}(d, q) = a * \text{sim}(d, q) + (1-a) * \text{PageRank} \quad 0 < a < 1 \qquad \text{or}$$

$$\text{final-sim}(d, q) = a * \text{inter-sim}(d, q) + (1-a) * \text{PageRank} \quad 0 < a < 1 \tag{4}$$

In the above expression, PageRank is normalized to a value between 0 and 1.

The BaseSet of documents are the top N documents obtained by formula (3), or re-ranking using proximity or formula (4).

2. Web TREC

2.1 Topic Distillation

"Topic distillation involves finding a list of key resources for a particular topic. A key resource is a page which, if someone built me a (short) list of key URLs in a topic area, I would like to see included." [TREC-2002Guidelines]

In TREC2002, we try to find the key resources by applying the following local link and URL analysis to the BaseSet.

Distillation Algorithm

In the BaseSet, the documents within each host are used to compute their qualities. This takes into considerations of:

a. The final similarity of each document within the host

b. The linkage information among the documents in the host.

(1) We first construct a graph as follows. A BaseSet of 1000 documents which have the largest similarities was obtained. This set is augmented by another set of documents S'. Each document, say d' in S' is a parent of a node d in S, and d' and d belong to the same host. Hyperlinks between documents on the same host form the directed edges of the graph.

(2) We now compute the quality of a document. It is essentially the sum of its final similarity and a weighted sum of the final similarities of its descendents within the same host. It is computed as follows.

Let N be the set of nodes in the parenthood graph; Let q[n] be the quality value of node n in N, and s[n] be its final similarity value.

a. Initialize q[n] to 0; s[n] = 0 if n does not belong to S, s[n] = its similarity value if n is in S.

b. For each node n in set N, compute its shortest path to each of its descendents, m. Let the length of the shortest path be d[n, m]. d[n, m] = 0 , if m = n.

c. q[n] = $$\sum_{\substack{m\in N \\ descendant\_of\_n}} \frac{s(m)}{2^{d(n,m)}}$$

(3) Rank the documents in descending order of quality. Return the 10 documents with the largest qualities, provided that no more than 1 document comes from the same host. In other words, if document d_1 and document d_2 come from the same host and d_1 has higher quality, then discard d_2.

Result

For topic distillation task, 4 BaseSets are constructed:

The first BaseSet contains documents retrieved using passage retrieval only.

The second BaseSet contains documents obtained using passage retrieval but with proximity take into consideration (section 1.3).

The third BaseSet contains documents obtained using passage retrieval, proximity and combined with PageRank.

The fourth BaseSet contains documents retrieved using document retrieval combined with PageRank.

Notice that this year's TREC topic distillation task only judges the top 10 retrieved documents. So in our submission, for each query, top 10 documents are from distillation, and the remaining documents are from BaseSet.

The best result is obtained by applying distillation algorithm to the third BaseSet, we get 52 key resources.

2.2 Named Page Finding

Surrogate Index and Retrieval

Since title and anchor text are very important in homepage finding, we treat them in a special way and generate separate index set of title and anchor text for afterward retrieving. The title and anchor text will be indexed as follows:

(1) For each HTML page p, extract its title T.

(2) For each HTML page p, construct a set S of anchor texts associated with the link in the Web pages that have link(s) pointing to p. For example, if page q has a link pointing to p, e.g. on page q there is <a> href=http://www.dbis.uic.edu>UIC database and information xxx Web site</a> and page p is dbis's web site. Then "UIC ..." is the anchor texts for page p. Note that it is extracted from page q.

(3) For each HTML page, construct a document surrogate from all the tokens in T and S.

(4) Index surrogates as we described in section 1.1.

For a given query each retrieved document will have a surrogate similarity. Then it is combined with the original document similarity to get the final similarity. The procedure is as follows:

(1) Compute the similarity of query q with surrogate: sim(q, surrogate-of-doc)

(2) Interpolate its similarity with the original document p, to get final similarity:

$$c * \text{sim}(p, q) + (1-c)* \text{sim}(\text{surrogate-of-p}, q) \qquad (5)$$

Result

We submit 3 runs for named page finding task.

The first run is obtained from the surrogate as described in formula (5).

The second run is obtained by combining document retrieval with PageRank.

The third rum is obtained using passage retrieval only.

The best is the third run, the average reciprocal over 150 topics is 0.564, the number of topics for which the named page found in top 10 is 114 (76%), the number of topics for which no named page was found is 20 (13.3%).

3. Conclusion

Our TREC-2002 experiments show that passage retrieval is beneficial to both topic distillation and named page finding. Proximity ranking can improve the precision of content retrieval. Linkage information (PageRank) helps both tasks. But unfortunately, the surrogate (title and anchor text) does not give us a good performance in named page finding task. Since this is the first time we participate in TREC, we do not have the time to try more sophisticated techniques. We are experimenting with new techniques to perform retrieval, which hopefully will yield much better effectiveness in the future.

Reference

[BrinPage98] Sergey Brin and Lawrence Page. The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30(1-7):107-117, 1998

[ChakrabartiDom99] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, S.R. Kumar, P. Raghavan, S. Rajagopalan and A. Tomkins, Mining the link structure of the World Wide web, *IEEE Computer*, 1999.

[Havelinwala99] Haveliwala, T. Efficient computation of pagerank. *Technical Report*, Stanford University, Stanford, CA., 1999

[PageBrinMotwani98] L. Page, S. Brin, R. Motwani, and Terry Winograd. The pagerank citation ranking: Bring order to the web. *Technical Report*, Stanford University, 1998

[RobertsonSparck76] S. Robertson and K. Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129--146, 1976.

[RobertsonWalker00] Robertson S E, Walker S. Okapi/Keenbow at TREC-8, *The Eight Text REtrieval Conference*, NIST SP 500-264, pp151-161, 2000

[SilvaRibeiro-Neto00] IR Silva, B Ribeiro-Neto, P Calado, N Ziviani, and ES Moura. Link-based and Content-based Evidential Information in a Belief Network Model. *Proceedings of the 23rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 96-103, 2000.

[SinghalKaszkiel01] Amit Singhal, Marcin Kaszkiel. A Case Study in Web Search Using TREC Algorithms. *Proceedings of the 10th International World Wide Web Conference*, pages 708-716, HongKong, May 2001

[TREC-2002Guidelines] TREC-2002 Web Track Guidelines, http://www.ted.cmis.csiro.au/TRECWeb/guidelines_2002, 2002

Question-Answering via Enhanced Understanding of Questions

Dan Roth
Chad Cumby, Xin Li, Paul Morie, Ramya Nagarajan,
Nick Rizzolo, Kevin Small, Wen-tau Yih
Department of Computer Science
University of Illinois at Urbana-Champaign

Abstract

We describe a machine learning centered approach to developing an open domain question answering system. The system was developed in the summer of 2002, building upon several existing machine learning based NLP modules developed within a unified framework.

Both queries and data were pre-processed and augmented with pos tagging, shallow parsing information, and some level of semantic categorization (beyond named entities) using a SNoW based machine learning approach. Given these as input, the system proceeds as an incremental constraint satisfaction process. A machine learning based question analysis module extracts structural and semantic constraints on the answer, including a fine classification of the desired answer type. The system continues in several steps to identify candidate passages and then extracts an answer that best satisfies the constraints.

With the available machine learning technologies, the system was developed in six weeks with the goal of identifying some of the key research issues of QA and challenges to it.

1 Introduction

The question answering track in TREC 2002 requires participants to construct a system that can answer open-domain natural language questions automatically given a large collection of news articles with the possible help of external knowledge sources. The questions are all factual and fairly restricted in the syntactic structure and the given answer should be a single phrase, without any additional information. Contrary to previous years, definition questions do not occur in this year's test set.

The open domain question answering (QA) system described in this paper has been implemented as a platform for studying and experimenting with a unified approach to learning, knowledge representation and inference that, we believe, is required to perform knowledge intensive natural language based inferences.

The fundamental assumption that underlies the system described here is that deep analysis of questions plays an important role in the question answering task. An information retrieval(IR) module, an answer selection and verification(AS) module and other supporting components of a QA system all rely on the information extracted from questions which we view here as *constraints* on possible answers. Our approach views the selection of the answer and its justification as an incremental constraint satisfaction process – information extracted from the question constrains the syntactical and semantical structures that can appear in corresponding answers. This process applies restrictions extracted by analyzing the questions using machine learning based classifiers and operates on data annotated also by machine learning based classifiers.

Specifically, the system analyzes the question to extract structural and semantic constraints on the answer and then proceeds in several steps, including a passage retrieval and an answer selection stage, to narrow down the list of candidate answers and rank them based on how well they satisfy the known constraints on the desired answer.

The system is centered around a unified machine learning and inference approach. Classifiers learned according to the SNoW learning architecture [2; 14] are used along with a SNoW based CSCL approach [12] to augment the questions and text documents with additional information including pos tagging information, shallow parsing and some level of semantic categorization (beyond named entities) - information that all the modules of our system exploit, including passage retrieval and answer selection. The same learning architecture is also used to train a question analysis module that provides an accurate and fine-grained semantic classification of the desired answer as well as deeper analysis, including identifying the required relation (if relevant) and some other syntactic and semantic constraints on the answer.

Consider the question [5]:

```
What is the fastest car in the world?
```

The candidate answers are:

1. `Jaguar.` With the justification: `...the Jaguar XJ220 is the dearest (415,000 pounds), fastest (217mph) and most sought after car in the world.`

2. `Volkswagen.` With the justification: `...will stretch Volkswagen's lead in the world's fastest growing vehicle market. Demand for cars is expected to soar.`

If we only consider a set of search terms — probably `fastest`, `car` and `world` — and assume that one knows that the search is for a proper name, then both sentences equally qualify as justifications. However, a slightly deeper analysis reveals that "fastest" needs to modify "car", while in one of the candidate sentences it modifies "market". In many cases, especially given the current definition of the TREC QA task, ad hoc proximity constraints can do the job. That is, key terms that are close together are more likely than not to be indicative of the correct answer. However, we believe that in order to make big progress in this task, this level of deeper analysis is still necessary.

The constraints may consist of various syntactic and semantic conditions that an answer has to, or is very likely to satisfy. For example, the answer may be known to be a noun phrase, a phrase that describes a location, a date or a city. If the answer is a person's name, an additional constraint may specify an action that this person is taking; It may specify a plurality or a gender constraint, etc.

Information extracted from the question analysis not only constrains possible answers, but also guides the downstream processing in choosing the most suitable techniques to deal with different types of questions. For example, given the question "*Who was the first woman killed in the Vietnam War?*", we want to avoid testing every noun phrase as an answer candidate. At the very least, we would like to be able to tell that the target of this question is a *person*, thereby reducing the space of possible answers significantly by only searching person names.

To achieve this goal, the QA system described here was constructed with an enhanced question analysis module. In the stage of question analysis, a fine-grained and fairly accurate question classification is performed to identify some of 50 predefined classes as the semantic classes of the desired answer. This is done using a machine learning technique that builds on an augmented representation of the question that includes pos tagging, shallow parsing information, and some semantic categorization information. A set of typical semantic relations is extracted from the questions, along with one of their arguments, and we also identify whether the desired answer is the first or the second argument of a binary relation. Eventually, the goal of question analysis is to generate an abstract representation of the question, based on syntactic and semantic analysis.

A passage retrieval module attempts first to apply the information extracted by the question analysis module to determining an ordered set of key terms (along with their properties, e.g., named entities) and construct a search policy to locate a passage that contains the answer. The passage retrieval module is based on the concept of structured text queries which are extended by searching over non-textual concepts such as named entities and part of speech types. Targeted strategies to generate queries for specific question types are formed according to the question analysis results with the help of additional information hinging, of course, on massive document preprocessing and indexing. Several interesting techniques have been developed for that purpose which allow, for example, this module to index different writings of a named entity with a single key.

Once a set of candidate passages are chosen, their location information is passed to the answer selection module to determine the most appropriate answer. The answer selection stage becomes more challenging due to the new requirements enforced in this year's TREC competition: the answers provided by the QA system should be the exact phrase of the answer; and, only a single answer can be output.

Our answer selection is thus an optimization procedure that is constructed, at this point, in an ad hoc way, with an attempt to increase precision. For this purpose, we focused only on answers that can be supported within a *single* sentence and otherwise we just returned "no answer" (NIL). The performance of our system on questions that have no answer is 48% and on questions whose answer is believed to be supported by the system, 24%. This is achieved by several methods. First, the IR module enforces key terms to be within a small window in the returned passages. The AS module itself implements strict constraint-satisfaction criteria in locating answers. In addition, by utilizing the relation information extracted by the question analyzer, it is possible to capture the right answers for specific types of questions and even rely on a knowledge base in some cases. Finally, a procedure for ranking answers is adopted to pick the most suitable answer from the candidates.

This project, with the exception of the IR module, started as a summer project in early June, 2002 and was completed before the end of July. The reliance on mature learning methods allowed us to put together a system for this task in such a short time. Needless to say, there are several important components that are still missing and many of our design goals have not been fully achieved.

Another working assumption is that a robust and accurate question answering system will depend on a large number of predictors. These will be used at many levels of the process and will support a variety of functions, from knowledge acquisition to decision making and integration of information sources. Along with these, there needs to be knowledge representation support that allows, for example, to keep track of predictions as input to higher level decisions and to maintain a coherent representation of a question or a story; and, there also needs to be an ability to make inferences by combining the outcomes of lower level predictions along with some constraints, e.g., those that are implied by the questions. Some of our modules already make use of this view by integrating different levels of classifications and inferences

[12; 15]. However, at the system level, the system developed here only makes some preliminary steps in these directions by putting forward a suggestion for a few of the learning components and a few of the higher level inferences required within a question answering system.

The rest of the paper is organized as follows: Sec. 2 describes the overall system architecture and highlights some of the methods applied. Sec. 3 summarizes the preprocessing of the queries and text. Sec. 4 describes the question analysis module. Sec. 5 and 6 present our information retrieval technique and answer selection and verification module. In Sec. 7, we provide preliminary evaluation on some of the system modules.

2 System Description

The three major modules of our QA system are the question analysis, information retrieval and answer selection modules. Each of them is a combination of multiple submodules and tools. The question analysis module extracts constraints from a question and stores them in a question analysis record. The IR module uses this information to extract relevant passages from the corresponding documents that are indexed at the word, sometimes the phrase and the named entity level. The answer selection module analyzes the candidate passages or searches a small knowledge base to extract the exact answer for the question. We highlight below a few important processing steps in our system.

1. An improved question analysis module provides an accurate question classification (answer semantic classes) and a detailed analysis of the question. The IR and the AS module utilize the analyzed questions and the answer semantic class to select query terms and locate a candidate answer, respectively.

2. A new indexing mechanism based on named entities and pos tags is applied to the document set and a passage retrieval module is employed, that makes use of the concept of structured text queries; it is also capable of searching over non-textual concepts such as named entities and part of speech types.

3. Question-specific strategies are applied both in the question analysis and in answer selection. In the question analysis they enable, for example, recognizing some binary semantic relations along with a missing argument, which is the target of the question. In answer selection, the semantic classes of the answer, as predicted by the question classifier, guide the application of class–specific answer selection rules.

4. A small knowledge base has been acquired and incorporated into the system. It contains a collection of binary relations along with their arguments and can be accessed using the name of the relation and one of the arguments, to extract the second argument. For example, the relation that A is the capital of B will be stored in our knowledge base for all countries and U.S. states. When the answer analyzer identifies that a question is concerned with this relation, the knowledge base will be searched, but after that, the normal AS process will still search for a justification for this answer.

Figure 1 presents the basic structure of our QA system. The following sections will give a detailed introduction of the internal modules.

3 Preprocessing of Questions and Text

All the questions and documents were preprocessed using a part–of–speech tagger, a named entity tagger and a shallow parser which perform a basic–level syntactic and semantic analysis.

The pos tagger is a SNoW based one [4] that makes use of a sequential model of classification to restrict the number of competing classes (pos tags) while maintaining, with high probability, the presence of the true outcome in the candidate set. The same method is used to give pos tags to both known and unknown words. Overall, as shown in [4], it achieves state–of–the–art results on this task and is significantly more efficient than other part-of-speech taggers. Note that we use the pos tagger both for the questions and the documents. Although pos taggers are typically evaluated on declarative sentences, we have evaluated our pos tagger also on questions and found its performance to be satisfactory.

Shallow parsing (text chunking) is the task of identifying phrases, possibly of several types, in natural language sentences. The shallow parser employed here is the SNoW based CSCL parser described in [13; 7]. Primitive classifiers are trained to identify the beginning and the end of each (type of) phrase. The final decision is made using a constraint satisfaction based inference, which takes into account constraints such as "Phrases do not overlap".

In question analysis, we apply this tool to identifying three types of phrases: noun-phrases, verb-phrases and prepositional-phrases. The definitions of these phrases follow those in the text chunking shared task in CoNLL-2000 [6]. When analyzing the documents, in order to save processing time, only noun-phrases and verb-phrases are identified. Below is an example to the type of information provided by this module.

Question: *Who was the first woman killed in the Vietnam War ?*
Chunking: *[NP Who] [VP was] [NP the first woman] [VP killed] [PP in] [NP the Vietnam War] ?*

Both the pos tagger and the shallow parser are available at `http://L2R.cs.uiuc.edu/~cogcomp`.

Our named entity recognizer categorizes noun phrases into one of 34 different categories of varying specificity. The scope of these categories is broader than usual for an average named entity recognizer. With additional categories such as **title**, **profession**, **event**, **holiday**, **festival**, **animal**, **plant**, **sport**, and **medical**, we redefine our task in the direction of semantic categorization. For the above example, the named entity tagger will get:

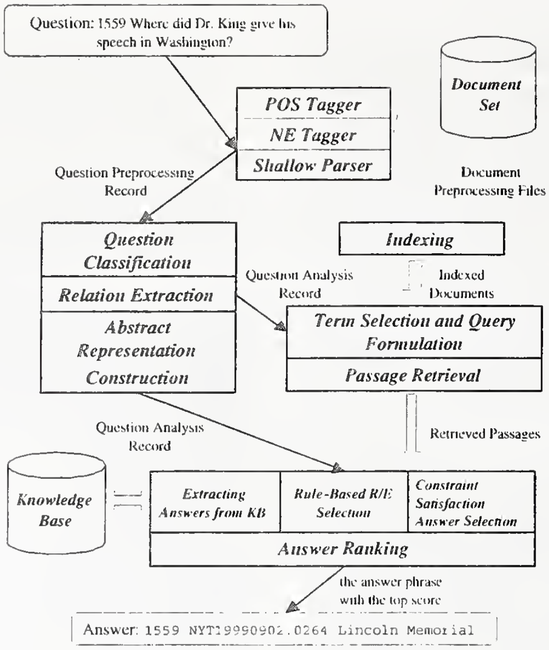

Question: 1559 Where did Dr. King give his speech in Washington?
POS Tagger
NE Tagger
Shallow Parser
Document Set
Question Preprocessing Record
Document Preprocessing Files
Question Classification
Relation Extraction
Abstract Representation Construction
Indexing
Question Analysis Record
Indexed Documents
Term Selection and Query Formulation
Passage Retrieval
Question Analysis Record
Retrieved Passages
Knowledge Base
Extracting Answers from KB
Rule-Based R/E Selection
Constraint Satisfaction Answer Selection
Answer Ranking
the answer phrase with the top score
Answer: 1559 NYT19990902.0264 Lincoln Memorial

Figure 1: System Architecture

NE: *Who was the [Num first] woman killed in the [Event Vietnam War] ?*

Like the shallow parser, the named entity recognition process centers around the SNoW based CSCL [13], with the addition of some predefined lists for some of the semantic categories. One major setback in developing this tool was a lack of sufficient training data. Since our decision process crucially depends on this categorization process, both in question classification and answer selection, we are planning to work on improving the accuracy of this tool.

4 Question Analysis

The goal of the question analysis is two-folded. First, we attempt to recognize the semantic type of the sought after answer, so that we can apply more specific and more accurate strategies when locating the answers. Second, we try to extract informative syntactic and semantic constraints over the possible answers.

To achieve these, question analysis consists of three subtasks. First, a machine learning approach is applied to performing a fine-grained question classification and identifying the semantic classes of the answer [8].

Second, given that the questions in TREC actually concentrate on a limited range of topics, it is possible to define a fairly small number of semantic relations that are the target of a large number of questions. Specifically, for a binary relation, the question identifies one of it's arguments, and looks for the other as the answer. In addition to the answer types, we therefore attempt to extract the target semantic relation in the question. For example, in the question "*When was Davy Crockett born ?*", the goal is to infer that the question is looking for the first argument in the relation *Birthdate_of (?, Davy Crockett)*. The information that the first argument is likely to be a date is also utilized.

Third, we fully parse the question and construct an abstract representation based on this parse result. The representation consists of some general syntactic and semantic relations such as subject–verb, verb–object, modifier–entity pairs, etc.

4.1 Fine Grained Question Classification

The purpose of question classification [8] is to identify the semantic classes of the desired answer. People working in QA seem to agree that this task is an essential and crucial step in the QA process. For example, [10] claims that 36.4% of the total errors of the QA system can be attributed to mistakes at this early stage. Moreover, it seems that the more specific the classification is, the greater the benefit to downstream processes. For example, in the next two questions, knowing that the targets are a "city" or a "country" will be more useful than just knowing that they are locations.

Q: What Canadian city has the largest population?

Q: Which country gave New York the Statue of Liberty?

Therefore, our taxonomy of question classification include 6 coarse classes(ABBREVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION and NUMERIC VALUE) and 50 fine classes (animal, color, event, food, language, plant, city, mountain, code, individual, title, speed, money, equivalent term, etc.). By using learning with multiple syntactic and semantic features, we can improve the classification accuracy to a level that can be relied upon in downstream processes. The question classifier developed is a two-layered hierarchical learned classifier based on the SNoW architecture. See [8] for details.

One difficulty in the question classification task is that there is no clear boundary between classes. Here are some examples of the internal ambiguity of this task.

Consider

1. What do bats eat?
2. What is the fear of lightning called ?
3. Who defeated the Spanish armada?

The answer of Question 1 could be food, plant or animal; The answer of Question 2 could be an equivalent term or a disease. And Question 3 could ask for a person or a country. Due to this ambiguity, it is hard to categorize these questions into one single class; it is likely that mistakes will propagate into any downstream processes. To avoid this problem, we allow the classifier to assign multiple class labels for a single question. This method is better than only allowing one label because we can apply all the classes in the downstream processing steps without loss of accuracy. In those steps, inaccurate answer candidates will be filtered out by applying further constraints over the answers. To implement this model, we choose to output k ($k \leq 5$) classes for a question. k here is decided by a decision module over the activation of each class. For example, for Question 2, all of *food, animal* and *plant* are returned as the possible answer types.

An important change in this year's TREC competition is that all definition questions are removed from the test set. This change increases the pressure on our question classifier because definition questions are relatively simpler to classify.

4.2 Relation Extraction

According to our statistics, about 30% of TREC 10 questions contained a specific and simple semantic relation which can be described as a binary relation. In TREC 2002, there are also a lot of relational questions. By extracting the semantic relations, the questions can be easily converted into a logical form. For example,

1. *When was Davy Crockett born ?*
   ⟶ Birthdate_of(?, Davy Crockett).
2. *What is the capital city of Algeria ?*
   ⟶ Capital_of(?, Algeria).
3. *Who invented the fishing reel ?*
   ⟶ Inventor_of(?, fishing reel).

The same relation should also be satisfied by the answer. Specifically, in these cases, the question specifies a binary relation along with one of its arguments, and the answer is the other argument. Several systems [1; 16] have previously shown that sometimes, even simple pattern matching methods can achieve high precision in answering those relational questions. While our goal is to develop a more general relation identifier [15], at this point, we define a set of over 30 specific binary semantic relations (such as **Birthdate_of, President_of, Stateflower_of, Inventor_of, Speed_of, Capital_of**, etc.) and apply heuristic rules to determining whether the questions satisfy typical patterns for those relations. For example, typical patterns to extract relations are like:

1. *Who invented/developed A ?* ⟶ Inventor_of(?, A)
2. *What's the capital [city] of A ?* ⟶ Capital_of(?,A)
3. *What's the [flying/...] speed of A ?* ⟶ Speed_of(?, A)
4. *How fast is A ? / does A fly ?* ⟶ Speed_of(?, A)

4.3 Abstract Representation

We represent questions using a simple abstract representation that reflects the basic dependencies between constituents in the question. The representation is extracted from a full parse tree of a question and contains the fields: *Action, Action Subject, Direct Object, Indirect Object, Target Description, Target Modifier, Action Modifier, Location, Time, Extreme Case, and Unit.*

5 Passage Retrieval using Structured Information Queries

In contrast to the more general method of passage extraction from a set of retrieved documents, our system directly retrieves candidate passages from the corpus for analysis by the answer selection module. We search for candidate passages along two primary dimensions: text structure and text classification. Along the structure dimension, our approach uses many of the concepts described within the frameworks of *overlapped lists* [3] and *proximal nodes* [11].

Constraining our search with structure, we are able to specify concept orderings and restrict the size of the text space that must contain the specified concepts. By searching over multiple text classifications, the simplest being the text itself, we are able to make more expressive restrictions over the corpus being searched. For example, in the question **How tall is the John Hancock Building?** searching for a document containing the terms **{john, hancock, building}** is not precisely searching for the desired information, but instead the most available information. When looking for the answer to this question, we only want occurrences of passages that explicitly or implicitly describe the John Hancock Building. A more appropriate description may be "a location near or containing the word **hancock**". This is due to the facts that the John Hancock Building is officially the John Hancock Center, people often simply refer to it as "the Hancock" in many contexts, and the words **{john, building}** are relatively common words, thereby providing limited information. However, we still want to eliminate references to John Hancock the person or John Hancock the company. While there are specific strategies to eliminate each of these problems in general information retrieval approaches, our system captures these ideas naturally and succinctly.

In this section, first we will briefly describe the indexing representation and searching mechanisms. Second, the query language will be provided including some examples of more common usages. Finally, we will describe the heuristics used to dynamically determine the retrieval query and determine when to return the list of feasible candidate passages.

5.1 Indexing and Searching Mechanisms

Indexing is performed via a set of document information files and a set of index term files. Each of these sets is comprised of dictionary files that are fixed size record files for hash lookup purposes and reverse index files which are of variable length per entry. Examples of files are stated below:

- *file.dict* contains a record for each document. Namely the fields of the entry are the document number, the document title, the starting position pointer of the document in the *file.struct* file, and the ending position pointer in the *file.struct* file.
- *file.struct* contains a list of sentence lengths for each document along with the length of the list (the number of sentences in the document). This can naturally be abstracted to other structural properties, but was not done in this case.
- *text.dict, NE.dict, ...* contains a record for each index term comprised of the index term, the starting position pointer in the *text.index* file and the corresponding ending position pointer.
- *text.index, NE.index, ...* contains a list of documents containing the index term that points to a list of *sentence*,*word* tuples representing all of the locations of that specific term.

Searches are performed in document number space and the results are translated into a <document, sentence range> pair for processing by the answer selection module. The fundamental search strategy is as follows. First, all documents containing the required search terms are extracted, much like in a Boolean retrieval model. The allowable range surrounding the first term in this set is calculated for each instance using the information contained in the *file.struct* file. Then each instance of the other terms is checked against the range calculated within the same document to see if it satisfies this constraint. In those cases that it does, the minimum window that contains all constraints is returned as a positive instance.

5.2 Query Language

When forming queries, the basic idea is that the first term serves as the initial set node and we search for a passage containing the subsequent terms within the proximity constraints in each direction. If only one proximity constraint is given, it is assumed to be symmetric. The *union* operator provides a method to induce a disjunction of structural and conceptual constraints. Table 1 shows this language in Backus-Naur form. One additional note is that the absense of a *concept* identifier when describing a term implies *text*, which is only syntactic sugar. This can be seen in the following examples:

```
query → term
      | ( operator query query {query} )
term → concept . keyword
concept → text | NE | POS | ···
operator → [ proximity ] | [ proximity proximity ]
      | [union]
proximity → ( integer , structure )
structure → document | sentence | word | ···
keyword → baseball | holyfield | nobel | ···
integer → ··· | − 1|0|1|2| ···
```

Table 1: Query Language BNF

1. Find the words *george* and *bush* in the same document. (standard boolean retrieval)
   ⟶ ([(0,document)] george bush)
2. Find the word *hancock* as part of a location.
   ⟶ ([(0,word)(1,word)] hancock NE.location)
   Note that this example also accounts for hancock being tagged incorrectly and building (or an equivalent word) being tagged correctly by the named entity recognizer.
3. Find the name *Snoop Dogg*.
   ⟶ ([(1,word)(2,word)] snoop dogg)
   Note that this will match *Snoop Dogg*, *Snoop Doggy Dogg*, and *Dogg, Snoop*.
4. Find passages describing the murder of John F. Kennedy.
   ⟶ ([(0,sentence)] ([(2,word)(3,word)] john kennedy) ([union] murder assasinate kill murdered assasinated killed))

While manual query generation seems somewhat cumbersone, queries are formulated using the information provided during question analysis which is the only interface to the human user. Therefore, this language captures the elements necessary for our constraint satisfaction approach in a succinct and easily usable form.

5.3 Term Selection and Query Formulation

While the query language and retrieval engine provide an expressive and efficient retrieval mechanism, the effectiveness is strictly limited by the queries generated. Our basic query formulation strategy is to iteratively select keywords and refine the query until a threshold quantity of passages is reached or refinement by additional terms would not return any documents that satisfies the required constraints. Since the answer selection mechanism was limited to a single sentence, the proximity constraint was always restricted to the value of *(0, sentence)*. Depending of the expected utility of the next keyword selected, additional operations such as term expansion or union operations are also performed. This process can be viewed as a greedy search over the space of document passages.

The first stage of the passage retrieval algorithm is to determine the strategy to follow based on results from question analysis. For each question analysis classification, an instance of the query formulation agent is instantiated. Each of these instances run independently and their results are accumulated and returned to the answer selector. Simple examples of basing the retrieval strategy on the question analysis module are if a proposed solution is found in the knowledge base or if a quotation is present in the question. In these cases, passages containing these concepts are first retrieved and additional keywords are used primarily to find support for the answer in the corpus.

In the general case, the aforementioned information is not available and we follow the basic search strategy by first extracting the two highest ranking keywords as scored according to Table 2. If two valid keywords cannot be extracted, we return to searching on a single keyword, but this is extremely rare and did not occur on any questions from the TREC contest. These keywords generate our initial search of the form:

([(0,sentence)] text.*keyword1* text.*keyword2*)

Description	Beginning Chunk Score	Default Score
Proper Nouns that are Named Entities	0	1
Nouns that are Named Entities	2	3
Remaining Named Entities except Adjectives or Adverbs	4	5
Remaining Named Entities	6	7
Extreme Adjectives	8	9
Nouns	10	11
Verbs	12	13
Any Remaining Keyword	14	14

Table 2: Ranking Keywords

Once the initial search is constructed, this set of passages is iteratively constrained by selecting the next highest ranking keyword until the termination condition is satisfied or until there are no usable keywords. The termination condition was tuned according to the equation $\frac{(passages_{max} - passages_{min})}{num\_score\_values} last\_score + passages_{min}$, where $last\_score$ is the score of the last keyword selected according to Table 2. In our case, $passages_{max} = 150$ and $passages_{min} = 25$ as determined experimentally. Once the number of candidate passages is less than this value, the results are passed to the answer selection module. The basic idea behind the termination condition is to continue refining the set of candidate passages if higher quality information is still available, but to stop this process once further refinement seems arbitrary. An example of this process follows.

When did Mike Tyson bite Holyfield's ear?
([(0,sentence)] tyson holyfield)
Result: 637 passages
([(0,sentence)] tyson holyfield ear)
Result: 99 passages
([(0,sentence)] tyson holyfield ear ([union] bite bit bites biting bitten burn burns burned prick pricked sting stings stung))
Result: 81 passages

Note that we did not search on the term *Mike* as it is a first name, which generally provides limited utility, and *81* passages seems a reasonable amount. Even though the approach of refining by desired named entity type was abandoned for the contest since it was not fully developed, the earliest experiment was to exploit the fact that we were searching for a date as an answer. Consider the case of the query,

([(0,sentence)] *previous result* NE.B-Date)
Result: 49 passages

which could be viewed as a evidence of a possibly promising approach. However, since the major effort was put into improving the general case, these approaches were not fully developed. Yet, similar strategies were developed to specifically look for words corresponding to abbreviations and abbreviations corresponding to words, but they are not discussed here since they were not used in the actual contest and are a subject of future study.

6 Answer Selection and Verification

Given question analysis records, the answer selection module locate and select the correct answer from extracted passages, taking the following three steps: Sentences in these passages are first analyzed syntactically and semantically by the pos tagger, the shallow parser, and the name entity recognizer. Candidate answers are then located in preprocessed passages. Because only the exact answer is judged as correct this year, we only consider specific types of named entities that the question asks for, or some basic noun phrases, as candidates. Finally, each candidate answer is evaluated and ranked. The top one is output as the final answer along with the document ID.

In the second step, different strategies are adopted to handle two different types of questions. If the question asks for an argument of a semantic relation identified, the answer selector first checks if our knowledge base has the answer. If exists, any occurrence of the answer string in the passages is simply picked. Otherwise, the module searches and identifies the relation in the passages and locate the sentences containing it. For the questions without a relation, the named entities or base noun phrases that satisfy other constraints identified by question analysis are chosen as candidates. Finally, these answers will be scored according to some heuristic rules.

6.1 Deriving Answers from the Knowledge Base

A part of the questions in TREC are asking for some simple facts, which can easily be answered with the help of a dictionary or an almanac. In addition, this type of information is usually available on the web. Examples of this type of question are *What is the capital city of Algeria?* or *Where is Lake Louise?* For the first one, our knowledge base stores the name of the capital city of each country. Once the *capital_of* relation is extracted from the question, we know the correct answer instantly. For the second one, our knowledge base stores the locations of many famous places in advance. Therefore, finding an answer in the document is just a simple string matching process.

Since in the TREC contest, we are required to not only find the answer but also return the document ID as the justification, randomly picking a document that has the answer string is not enough. If more than one document have the answer string, we (Section 6.4) will attempt to pick the document that has the best support.

6.2 Rule-based Relation/Entity Selection

In the rule-based portion of the answer selection module, we seek to apply pattern-based matching criteria to possible answer passages in order to identify relation pairs that correspond to the semantic relations observed in the question analysis. To match a potential candidate answer with a relation obtained from question analysis, we first determine which argument of the relation must be filled in by the candidate. That is, if we have identified a relation *Location_of* in the question and know Y corresponds to "the Statue of Liberty", which is located in X, then X is the argument which our candidate answer ought to match.

Therefore the next step in the matching process is to determine whether the Y argument obtained from question analysis is present in the sentence containing the candidate answer (candidate sentence). For each noun-phrase in the candidate sentence, we test whether it matches the known Y argument exactly or can be resolved to the same concept(meaning). If so we will proceed as outlined below. Otherwise we will test whether any noun-phrase in the candidate sentence matches the first noun-phrase of the Y argument. This partial matching is useful in cases where the argument identified tends to be very long, such as in the question *What is the name of the canopy at a Jewish wedding?* where the known argument of the identified *Name_Of* relation is *canopy at a Jewish wedding.*

Once the presence of Y is known in the candidate sentence, pattern matching rules based on relation types will be applied. We examine the words surrounding the positions of Y and the candidate answer. If they fit any pattern of the specified relation type, we add a predefined score to the total score of that candidate answer. Examples of the patterns for some common relations are listed below:

Location_Of:

1. Y `is (in,at)` ...X
2. X `(has,contains,includes)` ...Y

Capital_Of:

1. X `(is,became,was) ...the capital of` Y
2. `the capital of` Y `...(',',is,contains,includes)` X

For patterns partially matching the candidate sentence, the score awarded to the corresponding candidate answer is reduced compared to the score of full matching.

6.3 Constraint Satisfaction Answer Selection

Since the question analyzer provides detailed analysis of a question, the answer selection module only treats the named entities or base noun phrases that satisfy the constraints as candidate answers.

For example, if a question asks for a person name, then only the phrases that are tagged as *PERSON* will be considered. Note that there can be multiple answer semantic types for a question. Therefore, our refined constraint matchings are actually a mixture of decisions with the help of WordNet [9]. For instance, suppose a question asks for the name of the highest mountain in the US. Those phrases annotated as *LOCATION* are first picked by the named entity tagger as candidates. Then, we use WordNet to filter out those phrases that are not *mountains.*

WordNet can also help reduce the number of candidate answers when additional properties of the answer are given in the question. For questions like *What is a female rabbit called?*, WordNet can check if a string *is-a* rabbit.

Another example of constraint checking is the unit of a numeric answer. For instance, questions asking for *How long is Mississippi river?* cannot have an answer like "100 gallons".

6.4 Ranking Answers

To rank all the candidate answers, we evaluate the confidence with them based on heuristic rules. These rules generally test how closely the abstract representations of the candidate answers match the representation of the question. For instance, a candidate answer will get higher confidence if many of the nearby phrases contain or overlap *Target Modifier*, *Extreme Case*, or other semantic fields in the abstract representation identified from the question.

Note that, ideally, all answer candidates picked in the previous submodules should already be correct answers and ranked high by this submodule. However, since our relation extraction module is not 100 percent accurate and not every constraint derived from question analysis is fully checked, we just hope in this ranking process, the correct answer achieves a higher score than incorrect answers.

7 Evaluation and Error Analysis

Via an enhanced understanding of questions and other new techniques we incorporated with the QA system like the IR engine and new answer selection mechanisms, the total number of correct answers of our system has increased from last year's 54 out of 500(in rank 1) to 109 even with the stricter requirement over the answers(only the exact answer is counted as correct), which is an indication of the effectiveness of the recent work. What's more, the confidence-weighted score of the 500 answers reaches 0.299. Limited by the difficulty of creating reasonable and feasible evaluation standards for some modules of the question answering system, we could only perform evaluation results on part of our QA system. For the question classification and the information retrieval, we obtain evaluation results as follows:

7.1 Question Classification

The learning classifier for questions is built using a training set of 5,500 questions. Because of our specific decision model of allowing multiple labels for a question, in this paper, we count the number of correctly classified questions according to two different precision standards P_1 and $P_{\leq 5}$. Suppose k_i labels are output for the ith question after the decision model and are ordered decreasingly according to their density values computed by the learning algorithm in the classifier.

We define

$$I_{ij} = \{ \begin{array}{l} 1, \textit{ if the correct label of the ith} \\ \quad \textit{question is output in rank } j; \\ 0, \textit{ otherwise.} \end{array} \qquad (1)$$

Then, $P_1 = \sum_{i=1}^{m} I_{i1}/m$ and $P_{\leq 5} = \sum_{i=1}^{m} \sum_{j=1}^{k_i} I_{ij}/m$ where m is the total number of test examples. P_1 is the usual definition of precision which allows only one label for each question, while in $P_{\leq 5}$ we allow multiple labels.

We have a thorough evaluation of the question classifier on the 500 questions of TREC 10.

No.	Train	Test	P_1	$P_{<=5}$
1	1000	500	71.00	83.80
2	2000	500	77.80	88.20
3	3000	500	79.80	90.60
4	4000	500	80.00	91.20
5	5500	500	84.20	95.00

Table 3: Classification precision for fine classes on different training and test sets. Results are evaluated in P_1 and $P_{\leq 5}$.

For TREC 2002 question, our question classification accuracies reach 81% and 88.6% for the 50 fine classes in P_1 and $P_{\leq 5}$ separately. The average number of fine classes output for each question is only 2.15, which shows the decision model is accurate as well as efficient.

7.2 Information Retrieval

Tables 4 and 5 summarizes an evaluation of the passage retrieval module for the TREC 2002 contest questions, disregarding questions which are believed to have no answer contained in the corpus. Let P_j represent the percentage of questions that the passage retrieval module returns at least one passage containing justification for the correct answer. Let N_c be the number of passages returned for questions which generated at least one justified passage and N_i represent the number of passages returned for questions that did not generate any justified passages. These calculations were performed both on the singular sentences returned and on passages comprised of the singular sentence and a "window" of one sentence on each side. Table 4 shows the results when $passages_{max} = 150$ as done with our submission and table 5 shows the results when $passages_{max} = 300$ to show that constraining the number of allowable documents to a smaller threshold increases recall, but also increases the average number of passages returned.

	single sentence	one sentence window
P_j	59.7%	66.3%
N_c	54.2	51.9
N_i	26.0	25.1

Table 4: Passage Retrieval Evaluation: $passages_{max} = 150$

	single sentence	one sentence window
P_j	61.0%	68.7%
N_c	68.4	65.3
N_i	28.2	26.9

Table 5: Passage Retrieval Evaluation: $passsages_{max} = 300$

It should be noted that there were 7 questions that generated more than 500 passages and 8 questions that generated no passages, which were not considered in the above calculations. We also have evidence to support that recall percentages would further improve if the query strategy spanned multiple sentences, but as this approach was not used during TREC, we do not report these statistics here. Finally, it should be noted that the fact that the number of passages returned for questions that generate a justified passage is more than twice that of those that generate no such passage may point to a necessity for a more sophisticated search strategy beyond the greedy algorithm employed thus far.

8 Conclusion

TREC-like question answering requires generating some abstract representation of the question, extracting (for efficiency reasons) a small portion of relevant text and analyzing it to a level that allows matching it with the constraint imposed by the question. This process necessitates, we believe, learning a large number of classifiers, at several levels, that need to interact in various ways and be used as part of a reasoning process to yield the desired answer. Along with these, there needs to be a knowledge representation support that allows, for example, to keep track of predictions as input to higher level decisions and maintain a coherent representation of a question or a story; and, there needs to be an ability to use the outcomes of lower level predictions to make inferences that use several of these predictors along with some constraints, e.g., those that are implied by the questions.

This paper summarizes some preliminary steps we took in this direction by putting forward a suggestion for a few of the learning components and a few of the higher level inferences required within a question answering system. Some of our components work pretty well independently; however, at the system level, the system developed here hasn't achieved very satisfactory results,

This project, with the exception of the IR module, started as a summer project in early June, 2002 and was completed before the end of July. The reliance on mature learning methods allowed us to put together a system for this task in such a short time. Needless to say, there are several important components that are still missing and many of our design goals have not been fully developed. Some of our future research directions include developing our unified approach further in several directions. We plan to incorporate a level of semantic categorization that, hopefully, can impact all modules in the system; we are working on learning better representations for questions, incorporating inference across sentences in the answer selection process and a principled approach to answer selection.

References

[1] E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data-intensive question answering. In *Proceedings of Text REtrieval Conference (TREC-10)*, pages 393–400, 2001.

[2] A. Carlson, C. Cumby, J. Rosen, and D. Roth. The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department, May 1999.

[3] C. L. Clarke, G. V. Cormack, , and F. J. Burkowski. An algebra for structured text search and a framework for its implementation. *The Computer Journal*, 38(1):43–56, 1995.

[4] Y. Even-Zohar and D. Roth. A sequential model for multi class classification. In *EMNLP-2001, the SIGDAT Conference on Empirical Methods in Natural Language Processing*, pages 10–19, 2001.

[5] Sanda Harabagiu and Dan Moldovan. Open-domain textual question answering. In *Tuturial of the Second Meeting of the North American Chapter of the Association for Computational Linguistics*, 2001.

[6] E. F. T. Kim-Sang and S. Buchholz. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL-2000 and LLL-2000*, pages 127–132, 2000.

[7] X. Li and D. Roth. Exploring evidence for shallow parsing. In *Proc. of the Annual Conference on Computational Natural Language Learning*, 2001.

[8] X. Li and D. Roth. Learning question classifiers. In *COLING 2002, The 19th International Conference on Computational Linguistics*, pages 556–562, 2002.

[9] G. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K.J. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4):235–312, 1990.

[10] D. Moldovan, M. Pasca, S. Harabagiu, and M. Surdeanu. Performance issues and error analysis in an open-domain question answering system. In *Proceedings of the 40th Annual Meeting of ACL*, pages 33–40, 2002.

[11] G. Navarro and R. Baeza-Yates. Proximal nodes: A model to query document databases by content and structure. *ACM Transactions on Information Systems*, 15(4):400–435, 1997.

[12] V. Punyakanok and D. Roth. Shallow parsing by inferencing with classifiers. In *CoNLL*, pages 107–110, Lisbon, Protugal, 2000.

[13] V. Punyakanok and D. Roth. The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems*, pages 995–1001. MIT Press, 2001.

[14] D. Roth. Learning to resolve natural language ambiguities: A unified approach. In *Proc. of the American Association of Artificial Intelligence*, pages 806–813, 1998.

[15] D. Roth and W. Yih. Probabilistic reasoning for entity & relation recognition. In *COLING 2002, The 19th International Conference on Computational Linguistics*, pages 835–841, 2002.

[16] M.M. Soubbotin. Patterns of potential answer expressions as clues to the right. In *Proceedings of Text REtrieval Conference (TREC-10)*, pages 293–302, 2001.

Question Answering using the DLT System at TREC 2002

Richard F. E. Sutcliffe

Department of Computer Science
and Information Systems
University of Limerick
Limerick, Ireland

+353 61 202706 Tel
+353 61 202734 Fax
Richard.Sutcliffe@ul.ie Email
www.csis.ul.ie/staff/richard.sutcliffe URL

1. Introduction

This article outlines our participation in the Question Answering Track of the Text REtrieval Conference organised by the National Institute of Standards and Technology. Having not taken part before, our objective was to study the task and build a simple working system capable of answering at least some questions correctly. Only three person weeks was available for the work but this proved sufficient to achieve our goal. The article is structured as follows. Firstly, some preliminaries such as our starting point, tools and strategy are described. After this, the architecture of the Documents and Linguistic Technology Group's DLT system is outlined. Thirdly, the question types analysed by the system are described along with the named entities with which they work. Fourthly, the runs performed are presented together with the results we obtained. Finally, conclusions are drawn based on our findings.

2. Preliminaries

2.1 Previous Work

Our first step was to study previous attempts at the problem. Because of the limited time available, only two articles could be examined in detail. The first was Gaizauskas, Wakao, Humphreys, Cunningham and Wilks (1995). This describes the Sheffield LaSIE system and along the way provides many hints regarding techniques for effective information extraction and general text processing. The second was Rennert (2002). This excellent but unconventional article describes very concisely a number of key techniques used by the author in his 2001 TREC system. These ideas are highly pragmatic in nature and broken down by question type. In building our system we used Rennert's question taxonomy and methods as a starting point though in fact the final system was somewhat different as will be seen later.

2.2 Initial Tools

In computational terms, we started with some modest tools comprising a robust multiple pass parser (Sutcliffe, 2000) which we have used on a number of projects in Japanese (Sutcliffe and Nashimoto, 2000) and Spanish (Ruiz-Cascales, 2002) as well as English, a term recogniser and a general approach to natural language processing. To simplify the task we decided not to index the source documents ourselves but instead to use the TOPDOCS files provided by NIST and comprising the actual text of the top 50 matching documents found by the PRISE information retrieval system for each input query.

2.3 Overall Strategy

Beside the articles mentioned above we also devoted a small amount of time to looking at questions from previous years (mainly 2001) and establishing the relationship between them and answers in the corresponding documents. This led to an approximate strategy which has much in common with other QA systems and can be summarised as follows:

- Identify the type of the question;
- Based on the question type, search for appropriate named entities in the answer texts;
- Try to find a named entity which co-occurs with keywords from the question;
- Return the value of the 'best' such named entity as the answer.

3. Architecture of the System

3.1 Outline

We summarise here the architecture of the DLT system. Firstly, we identify the query type and hence the relevant named entities for which we will be searching. Secondly, we parse the 50 TOPDOCS Files dividing them into textual units using the markup. Thirdly, we search for instances of appropriate named entities in the textual units and mark them. Fourthly, we identify the winning named entity using one of two possible strategies: highest_scoring or most_frequent. We return this as the answer to the query. These stages are now dealt with in more detail.

3.2 Query Type Identification

Having identified the query types to use in the system, we studied questions of each type and developed simple keyword-based heuristics to recognise them. This is a very crude approach adopted due to shortage of time but it was suprisingly effective (Table 2): 425 of the 500 queries were correctly classified.

3.3 Text File Parsing

The text files are in XML-compliant form so it was easy to parse them without a Document Type Definition (DTD). Each document to be analysed was divided into a series of segments corresponding to a short passage of text. As with so many corpora, the level of markup varies from document to document and indeed we can not be sure that it has been used consistently. The strategy adopted was thus as follows: First, text within a HEADLINE tag was extracted. Second text within a TEXT tag was extracted and divided up into separate Ps. Finally a P was divided wherever three contiguous blanks were found. This last stage was to approximate sentence recognition. the resulting textual units were used in subsequent processing.

3.4 Named Entity Recognition

The type of question as identified in the first step determines the type of named entity or entities to be searched for. For example if the question type is what_state the entity is us_state i.e. 'California' etc. Each segment identified in the previous step was therefore inspected and all instances of appropriate named entities were identified.

Question Type	Example Question	Named Entities	Candidate Answer
state_bird	1517 What is the state bird of Alaska?	state_bird	Mockingbird
state_flower	1008* What is Hawaii's state flower?	state_flower	Yellow hibiscus
what_city	1520 What is the capital of Kentucky?	us_city, non_us_city	Houston
what_state	1743 Which state has the longest coastline on the Atlantic Ocean?	us_state	Calif.
what_county	1875 What county is St. Paul, Minnesota in?	us_county	Orange County
what_country	1496 What country is Berlin in?	country	Germany
what_continent	1489 What continent is India on?	continent	Asia
where	1500 Where is Georgetown University?	us_city, non_us_city us_state, us_county country	Phoenix, Ariz.
how_many3	1404 How many chromosomes does a human zygote have?	num	two chromosomes
how_much	1571 How much copper is in a penny?	num	20 percent
distance	1792 How far is it from Buffalo, New York to Syracuse, New York?	num, distance	100 miles
speed	1471 How fast does a cheetah run?	num, speed	60 miles an hour
temp	1606 What is the boiling point of water?	temp	212 degrees Fahrenheit
population	1750 What is Mexico's population?	num, population	one hundred million people
who	1395 Who is Tom Cruise married to?	proper_name	Nicole Kidman
when_interval	1056* When is hurricane season in the Caribbean?	date, interval	from June 1 to Nov. 30
length_of_time	1763 How old is the universe?	num, length_of_time	5 billion years
when	1698 When was Julius Caesar born?	date	100 B.C.
colour	1193* What color is a giraffe's tongue?	colour	black
unknown	1641 Where did 'N Sync get their name?	N/A	N/A

Table 1: Question Types used in the DLT system. The second column shows a sample question for each type. All are drawn from this year's data except for those question types which did not occur this year (indicated by an asterisk) where a sample from last year is shown. The third column lists the named entities which are used for answering a question of a particular type. The final column shows sample answers. These are all of appropriate types for the question but are not necessarily correct.

3.5 Answer Entity Selection

We experimented with two methods for selecting an answer which we call highest_scoring and most_frequent. In the first, we returned the named entity occurring in a textual unit which matched the keywords in the query best, chosen from any of the 50 PRISE documents. In the second, we returned the named entity which most frequently occurred in the vicinity of query keywords, observed across all occurrences of the entity in the 50 PRISE documents. Both strategies are unsophisticated but sometimes one or other of them can perform well on a particular query type.

In the next section we briefly outline the query types identified, the characteristics of the associated named entities and any special issues which affected processing for particular query forms.

Query Type	Classif.		Correct Classification								Incorrect Classification							
			Run 1				Run 2				Ru n 1				Run 1			
	C	NC	R	X	U	W	R	X	U	W	R	X	U	W	R	X	U	W
state_bird	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0
state_flower	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
what_city	13	0	2	0	1	10	2	0	1	10	0	0	0	0	0	0	0	0
what_state	1	5	0	0	0	1	0	0	0	1	1	0	0	4	1	0	0	4
what_county	3	0	0	0	0	3	0	0	0	3	0	0	0	0	0	0	0	0
what_country	8	1	2	0	0	6	2	0	0	6	0	0	0	1	0	0	0	1
what_continent	2	0	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0
where	40	3	11	1	1	27	7	0	1	32	0	0	0	3	0	0	0	3
how_many3	5	0	2	0	0	3	2	0	0	3	0	0	0	0	0	0	0	0
how_much	13	0	0	0	0	13	1	0	1	11	0	0	0	0	0	0	0	0
distance	21	0	4	0	0	17	4	0	0	17	0	0	0	0	0	0	0	0
speed	3	0	0	0	0	3	0	0	0	3	0	0	0	0	0	0	0	0
temp	2	0	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0
population	4	0	0	0	0	4	0	0	0	4	0	0	0	0	0	0	0	0
who	54	7	5	0	1	48	5	0	1	48	0	0	0	7	0	0	0	7
when_interval	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
length_of_time	7	0	1	0	0	6	1	0	1	5	0	0	0	0	0	0	0	0
when	92	6	15	1	1	75	15	1	1	75	0	0	0	6	0	0	0	6
colour	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
unknown	156	53	1	2	1	152	1	2	1	152	4	1	1	47	4	1	1	47
Totals	425	75	45	4	5	371	42	3	7	373	5	1	1	68	5	1	1	68

Table 2: Results by Query Type. The columns C and NC show the numbers of queries of a particular type which were classified correctly and not correctly. Those classified correctly are then broken down into Right, ineXact, Unsupported and Wrong for each of the two runs Run 1 and Run 2. Finally, those classified incorrectly are broken down in the same way.

4. Question Types and Corresponding Techniques

Given that time was short we were forced to build a basic system based on simple premises. We went about this by trying to identify the question types which occurred frequently in the 2001 question set and which could be answered using the above method. By the time the system had to be frozen for evaluation, there were twenty different question types as summarised in Table 1. Many more types could of course be added We outline here the techniques used for each one.

state_bird: Each state in the United States has its own state bird – a fascinating fact in itself made even more intriguing by the choice of the same bird by different states in several cases. A list of all such birds was readily drawn up. Any instances of these in documents could therefore be identified. This technique was hindered by two facts. Firstly, texts do not necessarily refer to a state bird by its official name – they might say Carolina Wren or even wren instead of Great Carolina Wren. Secondly, discussions about the bird for a particular state tend to occur close to mentions of other states and other state birds.

state_flower: The same strategy as for state birds was used, with the same limitations.

what_city: The TIPSTER Gazetteer Version 4.0 from the Consortium for Lexical Research (TIPSTER, 1992) was used to create a recogniser for place names. This gazetteer contains 246,907 entries of various kinds, mostly place names. Unfortunately our recogniser was too inefficient to use, so the list had to be reduced temporarily to a set of 199 US cities together with 167 capital cities for other counties. No heuristics were used for city recognition (e.g. X City is probably a city) due to lack of time. Some experiments with 'where' questions during system development suggested that the highest_scoring strategy was not very satisfactory for places because when one city is mentioned others can be also (e.g. in texts about air travel). This was the reason for developing the most_frequent strategy.

what_state: A list of all US states was readily obtained. Each name has three official forms (e.g. 'Massachusetts', 'Mass.' and 'MA'). Unofficial abbreviations are rare so the standard names are probably sufficient (compare this to state flowers and birds above). State names can appear in isolation or in combination with other units to form a location specifier (e.g. 'Boston, MA').

what_county: The TIPSTER Gazetteer contains a complete list of US counties but this proved too long to handle so a simple heuristic was used: Any name of the form X County was deemed to be a US county. This proved quite effective.

what_country: A list of 162 non-US countries was used together with a list of nine names for the US itself.

what_continent: A list of seven continents was used.

where: Templates were devised which specify the form of a place using various combinations of country, state, county and city. The longest possible specifier (i.e. the most complete one) was used whenever a candidate place was found in a text.

how_many3: A recogniser was developed for numbers by inspecting a large number of examples. Forms recognised include '1', '1.1', '1.1 million', 'sixty four', 'sixty four million'. Since tokenisation of the text only joined contiguous sequences of alphabetic characters and left all others separate, parsing numbers was fairly straightforward. Each was converted into a canonical representation for comparison purposes. Initially this was an integer in the case of whole numbers but interestingly some numbers found in the texts are so large that this caused integer overflow. A string representation was thus used instead. For how_many3 questions, the units are picked up from the query and must match those used in the document. For example in 'How many chromosomes' the units are 'chromosomes' and these must follow the number found in the text.

how_much: In these queries the units are not specified in the query and must be deduced from the text. The word following a number is accepted as the units if it is '$', '%' or another word longer than two characters which is not a number.

distance: A distance is a number followed by some distance units. Twelve plural and twelve singular distance units were collected by inspection of TREC texts.

speed: A speed is a number followed by some speed units. 21 plural and eleven singular speed units were collected from the texts.

temp: A temperature is a number followed by appropriate units with various premodifiers (e.g. minus. '-') and postmodifiers (e.g. 'above Absolute Zero'). Fifteen basic units, two premodifiers and eight postmodifiers were collected from the texts. Interestingly, 'degrees' with no explicit units implies Fahrenheit in a US text.

population: A population is considered to be any number of value one million or more which occurs in a text portion matching keywords from the query. The unit is assumed to be 'People'.

Query Type	Run 1 Harsh %	Run 1 Kind %	Run 2 Harsh %	Run 2 Kind %
state_bird	0	0	0	0
state_flower	0	0	0	0
what_city	15	15	15	15
what_state	0	0	0	0
what_county	0	0	0	0
what_country	22	25	22	25
what_continent	50	50	50	50
where	26	28	16	18
how_many3	40	40	40	40
how_much	0	0	8	8
distance	19	19	19	19
speed	0	0	0	0
temp	50	50	50	50
population	0	0	0	0
who	8	9	8	9
when_interval	0	0	0	0
length_of_time	14	14	14	14
when	15	16	15	16
colour	0	0	0	0
unknown	0	1	0	1
Totals	9	11	8	10

Table 3: Overall Performance. The harsh figure in each case indicates the percentage of queries which were classified as being of that particular type (either rightly or wrongly) and which were answered correctly. The kind figure indicates the percentage of queries which were rightly classified as being of that type and which were answered correctly.

who: To answer questions about names, a simple name recogniser was constructed. This allows pre-name titles (e.g. 'Sen.') a basic name sequence (e.g. Dwight G. Morgan) and a post-name title (e.g. 'III'). Given names are constrained to come from the MOBY given name word list (Ward, 1996). Originally the surname was drawn from the MOBY surnames but this proved too restrictive. Any capitalised word is thus accepted as a surname.

when_interval: These questions ask about a range of dates e.g. 'When is the hurricane season'. A recogniser was thus built which can handle 'from DATE1 through to DATE2' and so on.

length_of_time: A length of time is simply a number followed by one of nineteen different time units. Composed lengths (e.g. 4 min 33 sec) are not handled at present.

when: These imply the occurrence of a date in the text. Accordingly a recogniser for most of the US date forms was developed (e.g. 'Saturday, October 17, 1987' etc.). The value of a year specifier was constrained to be less than 10,000 if accompanied by 'AD' etc. (either before or after the date) thus allowing cases like '5,000 B. C.' On the other hand years with no 'AD' etc. were constrained to lie within the range 1700 and 2010. Dates in 'slash' or 'dot' formats (e.g. '02.06.02' or '02/06/02' to mean June 2, 2002 or perhaps February 6, 2002) are not handled.

Run	Qns	Time	Docs	Words	Characters	Min/Q	Words/Min	Words/Sec
Run 1	500	11h 0m	25,000	15,000,000	93,000,000	1.3	23,000	383
Run 2	500	7h 5m	25,000	15,000,000	93,000,000	0.85	35,294	588

Table 4: Timings and Counts for DLT. The columns indicate the identity of the run, the number of questions, the time to process all questions, the number of documents, words and characters processed, the time to process one query in minutes, and the numbers of words processed per minute and per second. See text for the system specification. The difference in timings for the two systems is caused merely by the fact that results of the SGML parsing were saved the first time around and therefore did not have to be done again.

colour: A list of nineteen colours was used. No doubt there are more but a longer list could not be found. In any case, no colours came up this year, only lists of colours.

unknown: This is the default query category. Approximately 156 queries fall into it. By definition these can not be answered by the system. As a crude experiment, all unknown queries were treated as if they were 'where', 'when' or 'who' in that order, accepting the first answer found.

5. Runs and Results

5.1 Two Experiments

We conducted two experiments each of which resulted in one run. All that varied in the runs was the method used for answer selection relative to each of the 20 query types. In the first run, what_continent, where, how_much and length_of_time were answered using the most_frequent strategy while the remainder were answered using highest_scoring. In the second run, all were answered using highest_scoring.

5.2 Results

Results are summarised in Tables 2 and 3. In Table 2, the two columns entitled 'Classif.' show the number of queries which were classified correctly (C) and incorrectly (NC) broken down by query type. Overall therefore, 425 of the 500 queries (85%) were categorised correctly. This seems quite a good result considering that the method used was based solely on identification of keywords and did not involve any structural analysis of the queries. By far the largest category of course is 'unknown' which accounts for approximately 156 queries (31%). We can conclude from this that 31% of the TREC 2002 questions fall completely out of the scope of this system. The remaining columns give counts of answers falling into the standard Right, ineXact, Unsupported and Wrong categories according to NIST assessors. The figures under 'Correct Classification' refer to queries which were correctly classified. Those under 'Incorrect Classification' refer to queries which should not be in the category. Very occasionally these were also answered correctly by chance.

Table 3 shows the overall performance of the system as a simple percentage of answers which were correct, broken down by question type. The Harsh figures take into account incorrectly classified queries as well as correctly classified ones. Clearly incorrectly classified queries will be answered incorrectly in the vast majority of cases, as shown by the last eight columns of Table 2. The Kind figures leave out incorrectly classified queries. We ignore the confidence rating of the system since we built in no mechanism for this. Dealing with Harsh figures, therefore, the overall performance of the system is 9% in Run 1 and 8% in Run 2. The best level of performance is for types 'what_continent' (50%), 'temp' (50%) and 'how_many3' (40%) but these are not representative because the number of each is small.

Perhaps the optimum view of the system is suggested by the figure of 26% Harsh (28% Kind) for 'where' queries in Run 1.

5.3 Timings

To give an indication of performance in terms of times, we used a Dell OptiPlex GX200 running NT4.0 at a clock speed of 733 MHz and having 256 Mb RAM. The whole system was written Quintus Prolog Release 3.4. Run times and related figures are provided in Table 4. Interestingly, the SGML parsing component is the slowest in the system and it is this which accounts for the difference in run times shown in the table. Speeding up this component therefore would be the most effective way of increasing performance.

6. Conclusions

Our objective was to get started in open domain question answering and to make as much progress in three weeks as possible. By the end of the time our system was up and running, giving a performance of 9% overall in Run 1. It has proved a useful experience and has helped us to focus on possible techniques for answering closed domain questions such as those analysed in Sutcliffe and Kurohashi (2000). There are of course many next steps to be taken which we summarise below:

- Analysing the query in more detail by parsing and hence identifying query type more accurately together with other information (e.g. the units in less simple cases such as 'How many pieces of clothing');
- Tidying up existing named entity recognisers and checking their performance;
- Adopting answer patterns rather than just using keyword co-occurrence, e.g. in the manner of Hovy et al. (2002);
- Determining other simple query types which account for the majority of the queries currently classified as unknown;
- Designing a strategy for identifying queries which have no answers;
- Developing the ability to recognise ad hoc named entities. For example in 'What type of car is used by Linda Ronstadt' we need to know all the car types despite having no pre-planned recogniser for them;
- Assigning a confidence rating to responses.

To finish on a more general note, two issues came to mind during this research. Firstly, a general theme within question answering seems to be the development of increasingly sophisticated pattern matching devices based on more and more detailed classifications of questions. To what extent however, will this lead to general findings about question answering or natural language processing rather than more and more brittle sets of rules?

Secondly, in TREC we can not always predict what types of question might be asked in the next competition. There were a few surprises this year, two examples being 1641 'Where did 'N Sync get their name?' and 1710 'What are the colors of the Italian flag?'. The first question is open ended in scope while the second involves lists of objects which we had assumed would be restricted to the list sub-track. Perhaps it would be fair for the general characteristics of new query types to be indicated so that some kind of strategy could be worked out for them.

8. References

Gaizauskas, R., Wakao, T., Humphreys, K., Cunningham, H., & Wilks, Y. (1995). University of Sheffield: Description of the LaSIE System as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference (MUC-6)* (pp. 207-220). Morgan Kaufmann.

Hovy, E. et al. (2002). *A Typology of over 140 Question-Answer Types.* Accessed 2002. http://www.isi.edu/natural-language/projects/webclopedia/Taxonomy/taxonomy_toplevel.html

Rennert, P. (2002). Word proximity QA system. In E. M. Voorhees and D. K. Harman (Eds.) *Proceedings of the Tenth Text REtrieval Conference (TREC 2001)* (pp. 179-191). Gaithersburg, Maryland: Department of Commerce, National Institute of Standards and Technology, NIST Special Publication 500-250.

Ruiz-Cascales, R. (2002). *A Specification and Validating Parser for Simplified Technical Spanish.* M.Sc. Thesis, University of Limerick, Ireland.

Sutcliffe, R. F. E. (2000). Using A Robust Layered Parser to Analyse Technical Manual Text. *Cuadernos de Filología Inglesa*, 9(1), 167-189. Número Monográfico: *Corpus-based Research in English Language and Linguistics*, University of Murcia, Spain.

Sutcliffe, R. F. E., & Kurohashi, S. (2000). A Parallel English-Japanese Query Collection for the Evaluation of On-Line Help Systems. *Proceedings of the Second International Conference on Language Resources and Evaluation, Athens, Greece, 31 May - 2 June, 2000*, 1665-1670.

Sutcliffe, R. F. E., & Nashimoto, N. (2000). Robust Parsing of Japanese Technical Text: An Initial Study. *Proceedings of the Eleventh Irish Conference on Artificial Intelligence and Cognitive Science, National University of Ireland Galway, 23-25 August, 2000.*

TIPSTER (1992). *Gazetteer 4.0.* ftp://crl.nmsu.edu/CLR/lexica/gazetteer/

Ward, G. (1996). *MOBY Project Wordlists.* ftp://ftp.dcs.shef.ac.uk/share/ilash/Moby

Novel Results and Some Answers

The University of Iowa TREC-11 Results

David Eichmann[1,2] and Padmini Srinivasan[1,3]

[1]School of Library and Information Science
[2]Computer Science Department
[3]Department of Management Sciences
The University of Iowa
{david-eichmann, padmini-srinivasan}@uiowa.edu

The University of Iowa participated in the novelty, adaptive filtering and question answering tracks of TREC-11. The filtering system used was an extension of the one used in TREC-7 [1] and TREC-8 [2]. Question answering was derived from the TREC-10 system. The novelty system was new…

1 – Novelty

With novelty being a new track, we had little prior experience to build upon. Hence we decided to begin our development experiments with a simple similarity match between the topic definition and the candidate sentence. For the available training topics, this proved to be remarkably responsive to tuning between precision-focused runs and recall-focused runs for novelty as well as the more predictable relevance decision. Figure 1 shows relevance tuning runs and Figure 2 new tuning runs. We used these runs as a baseline for our subsequent experiments in both threshold tuning

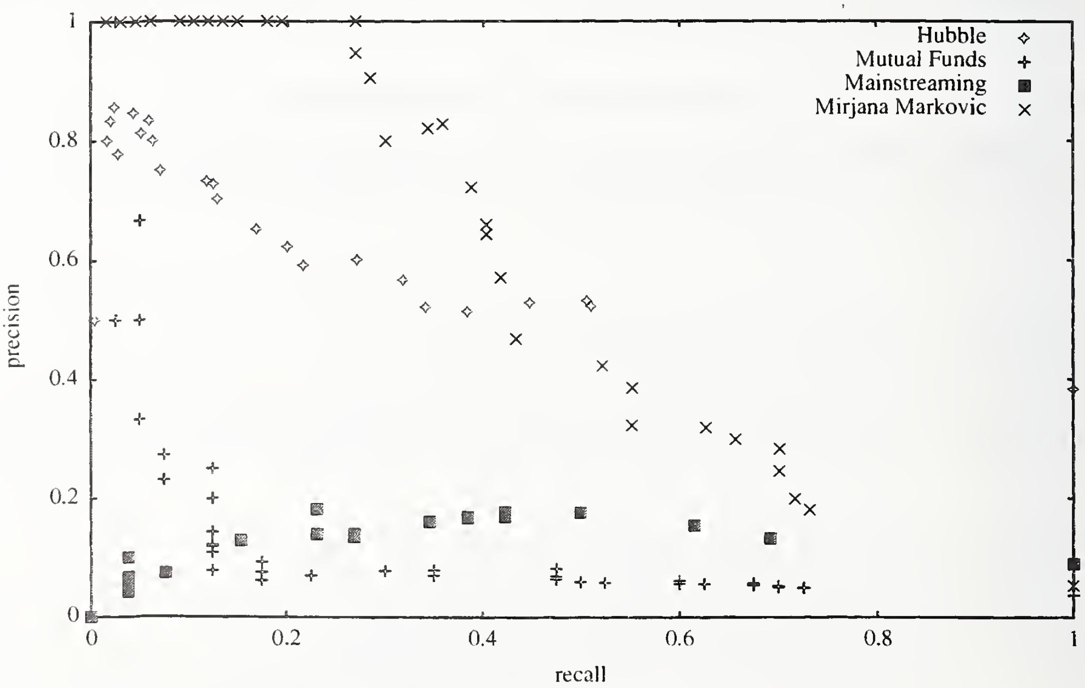


Figure 1: Training on Relevance, Simple Similarity

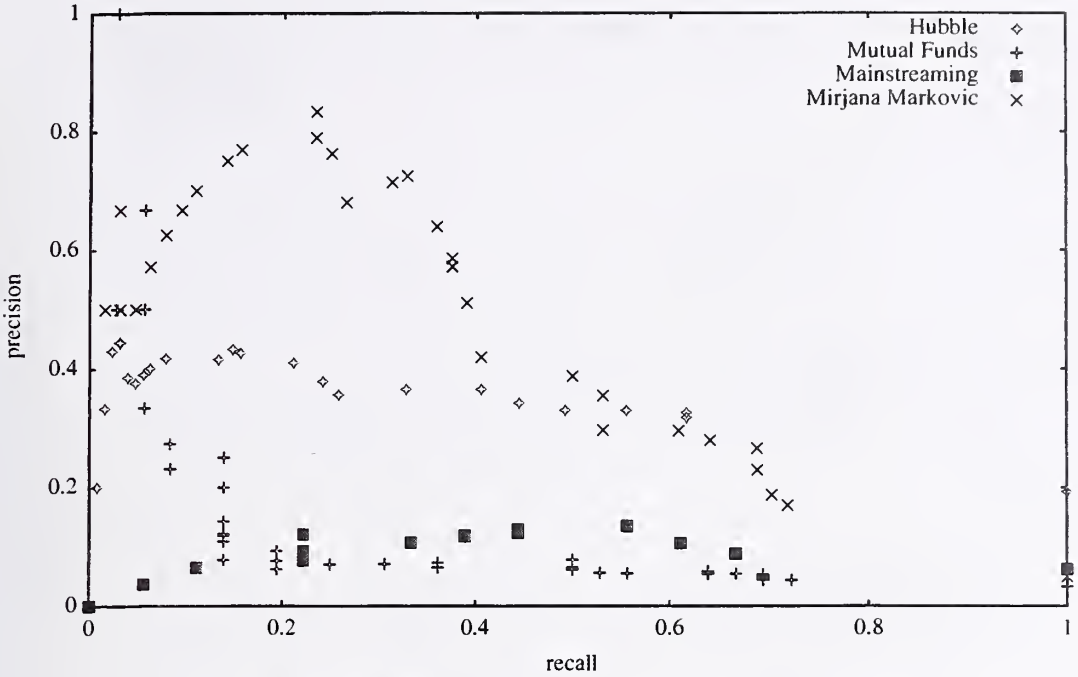


Figure 2: Training on New, Simple Similarity

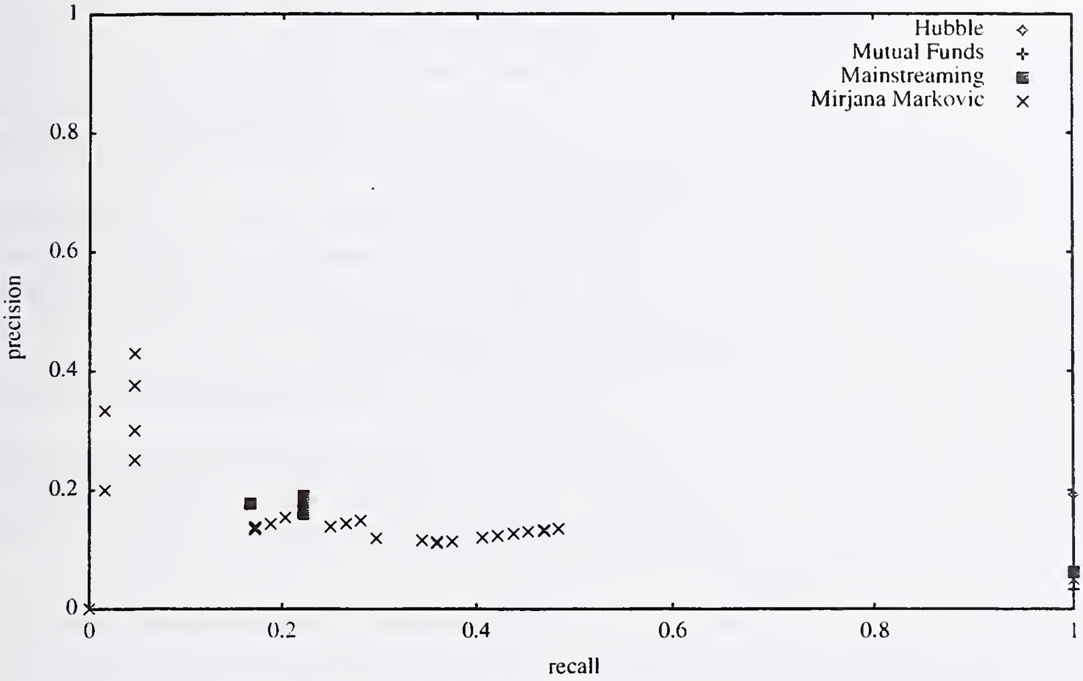


Figure 3: Training on New, New Entity Occurrence

and use of newly detected noun phrases and named entities. We first tried configurations that used single sources of similarity – e.g., just entities or just noun phrases. As shown in Figure 3 and Figure 4 for entities and noun phrases, respectively, this fared poorly

These preliminary experiments resulted in three different configurations:

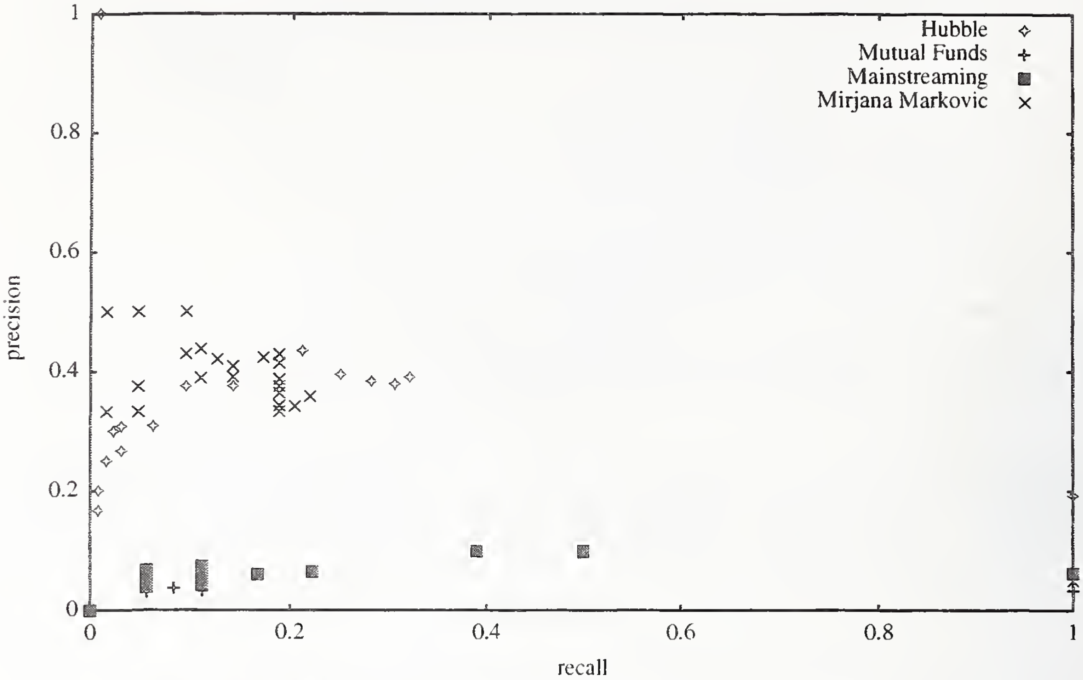


Figure 4: Training on New, New Noun Phrase Occurrence

1. simple stemmed term similarity, threshold-tunable for a range of precision/recall performance
2. new noun phrase triggering, guarded by a dual threshold of sentence similarity and full-document similarity. If the full document is sufficiently similar and the current sentence is sufficiently similar, the number of newly-detected noun phrases is compared against a minimum threshold and if the minimum is met, the current sentence is declared to be novel. Relevance in this configuration is driven by simple term similarity.
3. new named entity and noun phrase triggering, guarded by a dual threshold of sentence similarity and full-document similarity. If the full document is sufficiently similar and the current sentence is sufficiently similar, the number of newly-detected named entities and noun phrases is compared against a minimum threshold and if the minimum is met, the current sentence is declared to be novel. The named entities used in this configuration include persons, organizations and place names. Relevance in this configuration is driven by simple term similarity.

Configurations 2 and 3 hence are tunable both on the guard similarity thresholds and minimum threshold. Experiments with the training data indicated that a single new entity or noun phrase was sufficient for plausible performance against a range of guard thresholds for precision/recall tradeoff. Tuning runs on the configuration 3 yielded Figure 5.

Our official submissions included a run targeting a balance of precision and recall and a run targeting precision only. As shown in Tables 1 and 2, the runs ended up not that different in overall distribution of topic performance, but substantially better than random, as reported at the conference.*

* The track results reported elsewhere in this proceedings uses F = 2RP / (R+P), rather than F = R*P as used at the conference and in this paper.

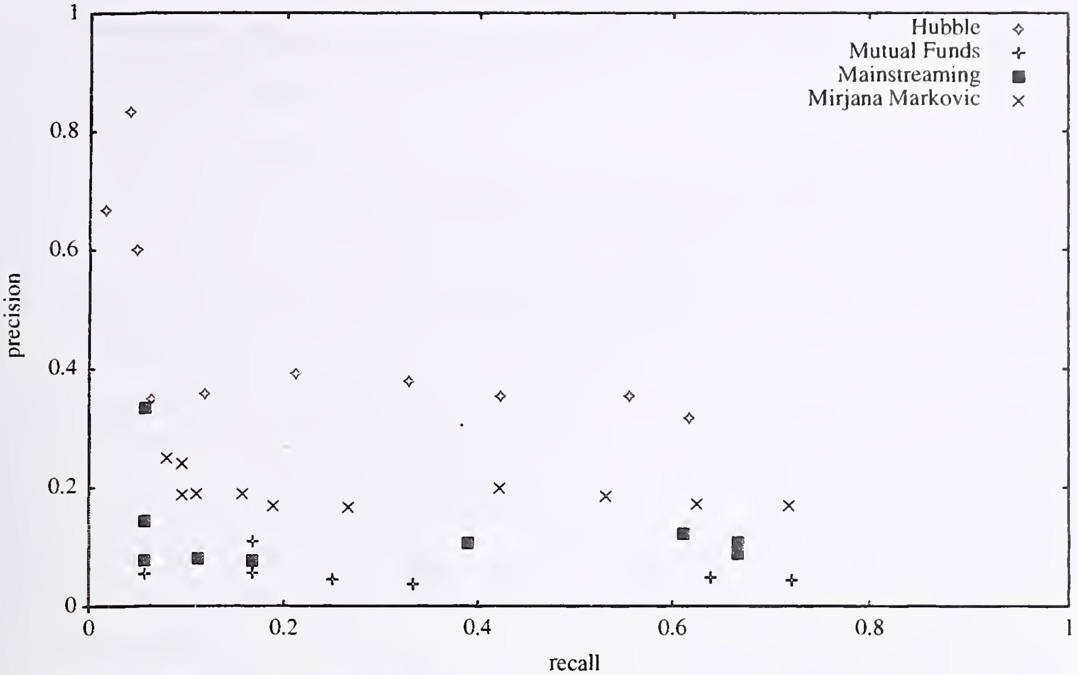


Figure 5: Training on New, New Entities & NPs with Guard

Table 1: Official Novelty Results, Relevant

	Precision & Recall Run	Precision Run	Human	Random
Ave. Prec.	0.12	0.14		
Ave. Recall	0.58	0.36		
Average P*R	0.073	0.059	0.191	0.006

Table 2: Official Novelty Results, New

	Precision & Recall Run	Precision Run	Human	Random
Ave. Prec.	0.12	0.14		
Ave. Recall	0.37	0.12		
Average P*R	0.048	0.020	0.170	0.004

Plotting the official runs' precision and recall, as shown for relevance in Figure 6 and novelty in Figure 7 shows that our tuning for 'precision' on the training data failed to improve precision on the test data to any noticeable degree, while substantially lowering recall. We plan to run further experiments to see if this is just a matter of 'failing to turn the knob sufficiently.'

Our overall impressions regarding novelty as an information retrieval task involve interesting inversions of expectations compared to relevance. For instance, in relevance, polysemy typically leads to false alarms and synonymy leads to misses. For novelty, our conjecture for further research

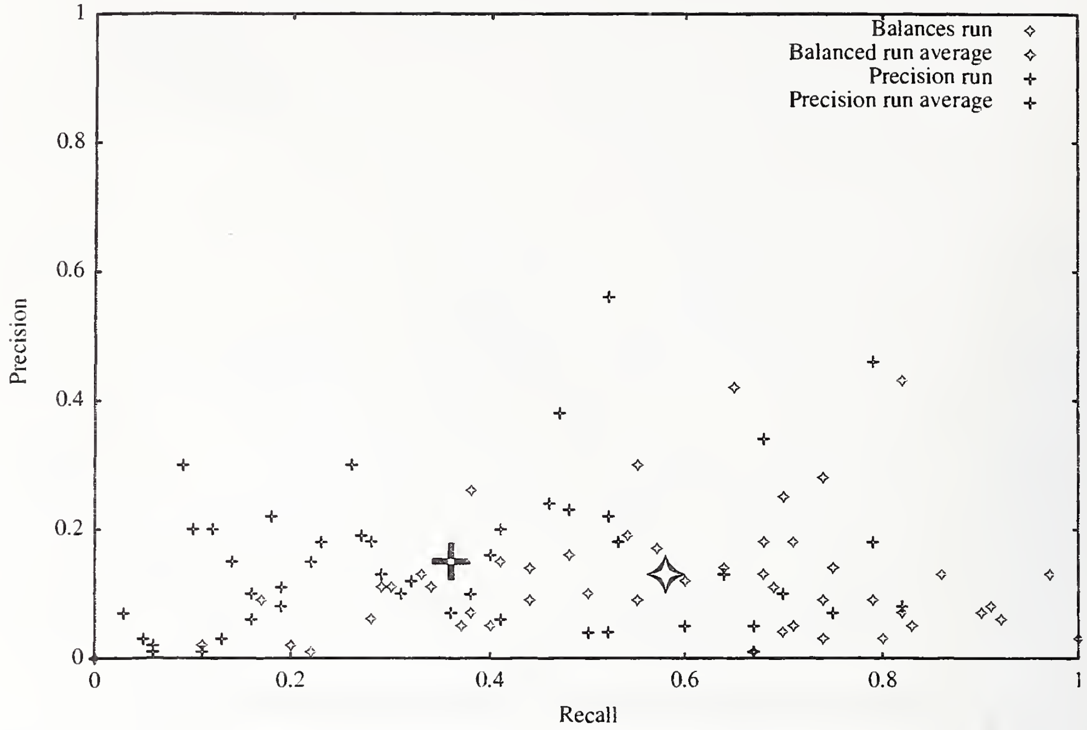


Figure 6: Official Relevance Runs, Precision vs. Recall, by Topic

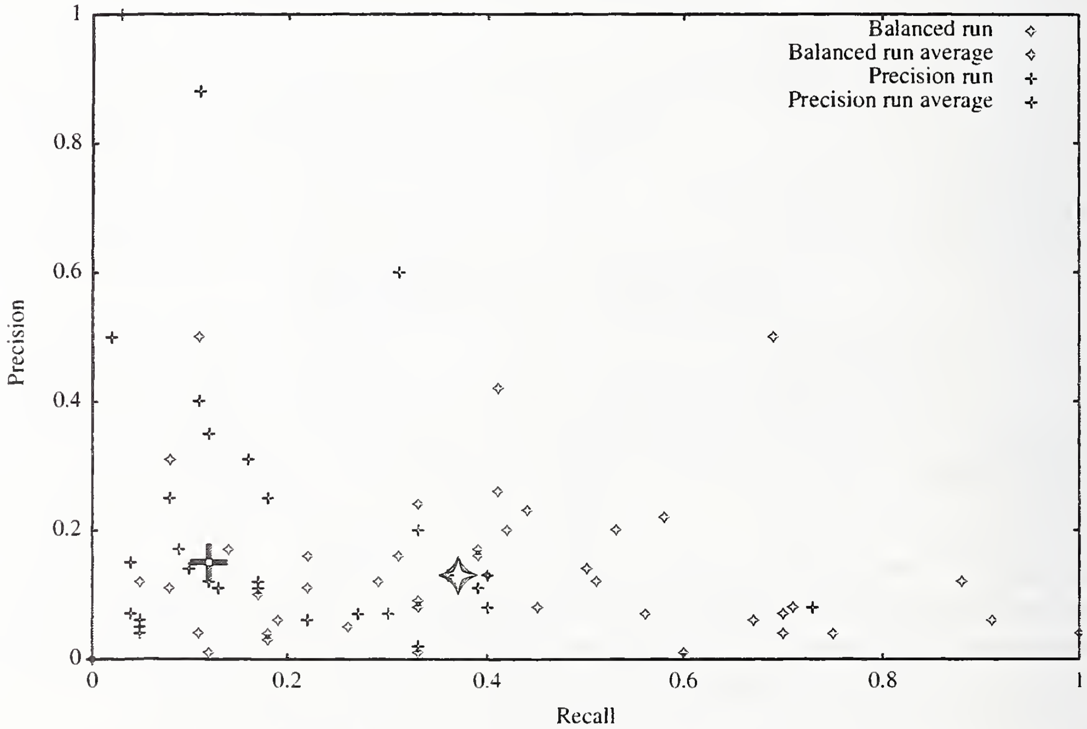


Figure 7: Official Novelty Runs, Precision vs. Recall, by Topic

involves polysemy leading to misses and synonymy leading to false alarms. Tuning our approach also clearly requires additional work, given the substantial differences in behavior between the training and test runs.

2 – Question Answering

Our system for QA is a complete overhaul of one of the two systems used for our previous TREC runs. We first used the previous year's questions to improve our classification of a question into one of a relatively small number of categories (quantity, person/organization, location, etc.) with testing achieving ~92% accuracy. Once the questions are classified, they are parsed using the CMU link grammar parser to extract subject/verb/object structure.

Each document in the corpus is then decomposed into doc-id / sentence pairs, with the sentence being the unit of analysis from that point. Each sentence is then POS-tagged and fed to the grammar parser. The parse tree for the sentence is then attributed with the POS tags for each word. Processing both queries and documents using this scheme allows us to establish both the nature of the query (using a fairly typical taxonomy) and the nature of the needed answer. This is particularly useful with respect to identification of candidate phrases in sentences and scoring of these phrases against the goal of the query.

The availability of the parse tree for the phrase allows for elision of subordinate clauses that can cause answers to span too long a string and for extraction of likely answers through heuristic matching of, for example, a subordinate clause immediately trailing a mention of a candidate named entity.

Table 3: Question Answering Results

	Best Single Score	Summed Scores	Response Frequency
Unsupported	0	2	4
Inexact	6	3	5
Right	18	21	15
Score	0.055	0.042	0.023

3 – Adaptive Filtering:

Our approach to filtering involves a two-level dynamic clustering technique. Each filtering topic is used to create a primary cluster that forms a general profile for the topic. Documents that are attracted into a primary cluster participate in a topic-specific second level clustering process yielding what we refer to as secondary clusters. These secondary clusters, depending upon their status, are responsible for declaring, i.e., retrieving, documents for the topic.

As documents are temporally processed they are attracted to a primary cluster if their similarity with the cluster vector is above a primary threshold. These documents enter the secondary clustering stage where again, based on similarity to cluster vectors and a secondary threshold, they either join an existing secondary cluster or start a new one. If at some point the similarity between a secondary cluster and the primary cluster exceeds a third declaration threshold then the document most recently added to the secondary cluster is retrieved for the user.

When deriving representations we use TF*IDF weights after stemming the terms using Porter's stemmer. We also limit document vectors and cluster vectors to the best 100 and 200 stems respectively.

References

[1] Eichmann, D., M. E. Ruiz and P. Srinivasan, "Cluster-Based Filtering for Adaptive and Batch Tasks," *Seventh Conference on Text Retrieval,* NIST, Washington, D.C., November 11 - 13, 1998.

[2] Eichmann, D. and P. Srinivasan, "Filters, Webs and Answers: The University of Iowa TREC-8 Results," *Eighth Conference on Text Retrieval,* NIST, Washington, D.C., November, 1999.

CARROT II and the TREC 11 Web Track

R. Scott Cost, Srikanth Kallurkar, Hemali Majithia, Charles Nicholas, and Yongmei Shi

University of Maryland, Baltimore County
Baltimore, MD USA
{cost,skallu1,hemal,nicholas,yshi1}@csee.umbc.edu

Abstract. We describe CARROT II, an agent-based architecture for distributed information retrieval and document collection management. CARROT II consists of an arbitrary number of agents, distributed across a variety of platforms and locations. CARROT II agents provide search services over local document collections or information sources. They advertise content-derived metadata that describes their local document store. This metadata is sent to other CARROT II agents which agree to act as brokers for that collection. and every agent in the system has the ability to serve as such a broker. A query can be sent to any CARROT II agent. which can decide to answer the query itself from its local collection. or to send the query on to other agents whose metadata indicate that they would be able to answer the query, or send the query on further. Search results from multiple agents are merged and returned to the user. CARROT II differs from similar systems in that metadata takes the form of an automatically generated, unstructured feature vector. and that any agent in the system can act as a broker. so there is no centralized control. We present experimental results of retrieval performance and effectiveness in a distributed environment.
We have evaluated CARROT II in the context of the Web Track of NIST's annual Text Retrieval Conference. Our methodology is described. and results are presented. [1]

1 Introduction

We have developed a scalable, distributed query routing and information retrieval system, CARROT II. It is the successor of an earlier project (Collaborative Agent-based Routing and Retrieval of Text) [8] (originally CAFE [7]). CARROT II is composed of a flexible hierarchy of query routing agents. These agents communicate with one another using KQML [9] and the Jackal platform [5]. and may be distributed across the Internet. While all agents in the system are alike, they can each control widely varying information systems. Agents interact with information sources via a well-defined interface. Queries presented to any agent in the system are routed, based on the content of the query and metadata about the contents of the servers. to the appropriate destination. Agents themselves are uniform and extremely simple.

CARROT II contains wrappers that extend other well-known IR systems (e.g. MG [19], Telltale [15]), as well as CARROT II's own, modest IR system. These wrappers present

[1] This document is an expanded version of a paper presented at the Workshop on Cooperative Information Agents Workshop (CIA '02) [6].

a basic interface to the CARROT II system for operating on documents and metadata. Agents are addressable via commands that are communicated in KQML. This means that a CARROT II system can be created, configured, and accessed by another information system, and so can be employed to extend the search capabilities of an existing project. We use the Jackal platform to support communication among agents in CARROT II and to provide an interface to the outside world.

Section 2 discusses the problems areas facing DIR and the efforts so far by the research community, Section 3 describes the CARROT II architecture and Section 4 describes the operation of the system. In Section 5, we describe our experience with TREC data.

2 Related Work

In the past there have been attempts to introduce the concepts of agent-based information retrieval. Systems like SavvySearch [13] demonstrated a simple approach to querying web search engines and combining their results in a single ranked order.

Historically, Harvest was the first system to demonstrate the use of broker agents in distributed search. The Harvest system [2] is a distributed, brokered system originally designed for searching Web, FTP, and Gopher servers. In Harvest, "gatherer" agents collect metadata from information providers and deliver it to one or more brokers. Metadata objects are represented in Summary Object Interchange Format (SOIF), an extensible attribute-value-based description record. Harvest pioneered the ideas of brokering, metasearch, replication, and caching on the Internet.

2.1 Distributed Information Retrieval

Information Retrieval in a distributed environment normally follows three steps [3]:

1. Information Source Selection: Select the best information source(s) per query
2. Query Processing: Send the query to the source(s) and return ranked list(s) of documents
3. Results Fusion: Create a single ranked list from ranked lists returned from the sources.

For retrieval from text, one of the methods for information source selection is use of automatically generated metadata from the content. Comparing the query to metadata about the sources can reveal the possible relevance of each source to the query. CORI [4] and gGloss [12] are examples of such metadata in information source selection. The CORI model is based on inference networks. CORI creates a virtual document containing Document Frequency (DF) and Inverse Collection Frequency (ICF). The ICF indicates the importance of the term across the collections and is analogous to the Inverse Document Frequency (IDF), which is a measure of term importance in a single collection. gGloss creates a virtual document containing DF(s) and Term Frequency (TF), i.e. number of occurrences per document of unique terms of the collection. French et al. [10] showed that CORI performed better than gGloss in terms of retrieval effectiveness, however they could not provide a reason for CORI's superior performance.

Gibbins and Hall [11] modeled query routing topologies for Resource Discovery in mediator based distributed information systems. Liu [14] demonstrated query routing using processes of query refinement and source selection, which interleaved query and database source profiles to obtain a subset of good databases.

The final step in answering a query is fusing the ranked lists from the queried sources to obtain a single ranked list. Voorhees et al. [18] used query training to determine the number of documents that should be selected from each source, then used a die based approach to determine the next source from which a document should be selected to form the final ranked list. Aslam and Montague [1] showed that results fusion based on ranks alone can be as good as regular fusion techniques and that relevance scores are not required.

In general, there is a performance gain by distributing information, but distributed retrieval lags behind centralized retrieval in terms of retrieval effectiveness, i.e. percentage of relevant documents returned for a query. However Powell et al. [16] showed that a distributed search can outperform a centralized search under certain conditions.

3 CARROT II Architecture

A CARROT II system is a collection of coordinated, distributed agents managing a set of possibly heterogeneous IR resources. These agents each perform the basic tasks of collection selection, query routing, query processing, and data fusion. In order to effect this coordination, some amount of underlying structure is required.

There are three components to the CARROT II architecture. The central work of CARROT II is performed by a distributed collection of CARROT II agents. There is also a network of infrastructure agents which facilitate communication and control of the system. Finally, a small set of support agents facilitates access to the system, and coordinates its activities. Each of these components is described in detail below.

3.1 CARROT II Agents

The CARROT II agent is the cornerstone of the CARROT II system. Its role is to manage a certain corpus, accept queries, and either answer them itself, or forward them to other CARROT II agents in the system. In order to do this, each CARROT II agent creates and distributes metadata describing its own corpus to other CARROT II agents. All CARROT II agents are identical, although the information systems they manage may vary.

A standard interface provides methods for manipulating documents, metadata, and handling queries. The agent maintains a catalog of metadata which contains information about the documents stored by peer agents.

3.2 Information Integration

A CARROT II agent interfaces with an information source which may be an ordinary IR package, or a database manager.

The CARROT II system currently has a wrapper that interfaces with the WONDIR[2] IR engine. It can however be extended to support other types of Information sources. As mentioned earlier, metadata is derived from the document collection. The metadata takes the form of a vector of the unique "N-grams" of the collection and a sum of their number of occurrences across all documents in the collection. Hence, unlike Harvest, CARROT II metadata describes the agent's collection of documents, not a single document. CARROT II uses such metadata for source selection. The motivation for such a form of metadata comes from relative ease of use, low cost of generation, and the ability to aggregate metadata, such that a single vector may contain metadata about multiple agents.

The same query operation can now be performed on both documents and metadata. A query operation returns a similarity score using $TF * IDF$ based cosine similarity [17]. Querying a collection returns a ranked list of the documents sorted by their similarity scores. For querying metadata collection IDF is replaced by the ICF (see Section 2).

3.3 CARROT II Infrastructure Agents

In order to support the successful inter-operation of potentially very many CARROT II agents, we have constructed a hierarchical infrastructure. The infrastructure is largely dormant while CARROT II is in operation, serving primarily to facilitate the orderly startup and shutdown of the system, and provide communications support.

The infrastructure is controlled by a single Master Agent, which may be located anywhere on the network. At startup, the Master Agent is instructed as to the number of agents required, and some factors regarding their desired distribution. These include the number of physical nodes to be used, as well as the degree of resource sharing at various levels.

The Master Agent starts a Node Agent on each participating machine, and delegates to it the task of creating a subset of the required agents. The node will be divided into Platforms, or independent Java Virtual Machines, each governed by a Platform Agent. The Node Agent creates an appropriate number of Platforms, and again delegates the creation of a set of agents.

Within each Platform, the Platform Agent creates a set of Cluster agents. The purpose of the Cluster Agent is to consolidate some of the 'heavier' resources that will be used by the CARROT II agents. Primarily, this means communication resources. A Cluster Agent maintains a single instance of Jackal. Each Cluster Agent creates a series of CARROT II agents; these run as subthreads of the Cluster Agent. Because most agents will be dormant at any one time, we allow a CARROT II agent to be assigned more than one collection, creating a set of 'virtual' agents. Thus the virtual agents are the agents visible to all entities external to the system.

[2] Word or N-gram based Dynamic Information Retrieval: an in-house system, developed as part of the CARROT II project

3.4 CARROT II Support Agents

While the agents in the CARROT II system work largely independently, a small set of support agents serve to coordinate the system's activities. These are the Agent Name Server, the Logger Agent, and the Collection Manager.

An Agent Name Server provides basic communication facilitation among the agents. Through the use of Jackal, CARROT II employs a hierarchical naming scheme, and its operation is distributed through a hierarchy of name servers.

A Logger Agent monitors log traffic, and allows the system to assemble information about the details of operation. This information can then be used to feed monitors or visualization tools.

Finally, a Collection Manager facilitates the distribution of data and metadata. It determines which collection of documents or information source will be assigned to each agent, how each agent will distribute its metadata, and what set of agents will be visible outside the system.

4 CARROT II Operation

Metadata distribution and query processing are the two main functions of the CARROT II agents. Recall from Section 3.2 that the CARROT II metadata is an automatically generated feature vector derived from the content itself.

4.1 Metadata Distribution

CARROT II uses a vector-based representation of metadata which describes the contents of the local corpus (see [8]). Metadata, as well as corpus documents, are managed by the IR engine (for example, WONDIR). The distribution of metadata has a profound impact on the system's ability to route queries effectively, and determines the "shape" of the CARROT II system. Agents receive instructions on metadata distribution from the Collection Manager. There are three metadata distribution modes that can be used by CARROT II:

1. Flood: Each agent broadcasts it's metadata to every other agent in the system. Under this scheme, any agent receiving a query would have complete (and identical) knowledge of the system, and be able in theory to find the optimal target for that query.
2. Global: As in the original CARROT architecture a designated broker agent has knowledge of the entire system. All agents share their metadata with only this agent.
3. Group: Once the system reaches a certain size, however, both of the above schemes are impractical; the system would become susceptible to bottlenecking. The group scheme is based on mathematical, quorum-based distributions, where a group of agents is represented by a chosen (or elected) agent. Metadata sharing would occur inside such groups and amongst the group leaders.

Metadata distribution can be effected by transferring the entire vector, a "difference" vector in case of changes in the agent's corpus, or just a URL pointing to the location of the metadata, enclosed in a KQML message.

4.2 Query Processing

Once the system is running, the Collection Manager becomes the primary or initial interface for outside clients. A client first contacts the Collection Manager to get the name or names of CARROT II agents that it may query. The names in this set will be determined by the metadata distribution policy. For example, in the case of group-based distribution, the set will contain the group leader agent names. The client then sends queries to randomly selected agents from the given set. It is also possible to model more restricted or brokered architectures by limiting the list to only one or a few agents, which would then feed queries to the remainder of the system.

In response to an incoming query, an agent decides whether the query should be answered locally, forwarded to other agents, or both. The agent compares the query to its local metadata collection and determines the best destinations. Based on the results, it may send the query to the single best source of information, or it may choose to send it to several. One of those sources may be its own local IR engine. Once answers are computed and received, the results are forwarded back to the originator of the query. If more than one information source is targeted, the agent faces the problem of fusing the information it receives into one coherent response.

Queries may be routed through a number of different agents before finally being resolved; this depends on the scheme used for metadata distribution and the routing algorithm. For example, the simplest scheme is to have each agent broadcast its metadata to every other agent in the system. The corresponding routing algorithm would be to route to the best information source. Since all agents have the same metadata collection and employ the same algorithm, a query will be forwarded at most once before being resolved. For schemes which employ a more efficient distribution of metadata, or possibly higher order metadata, queries may pass through many CARROT II agents before finally being resolved.

4.3 Implementation

The current implementation of CARROT II uses the flood mode of metadata distribution. This implies that a query given to any agent in the system will return the samc answer. The query can be routed either based on a metadata similarity cutoff or to the best N agents, based on the metadata similarity scores. Therefore, as stated in Section 4.2, the query needs to be *forwarded* only once. The results fusion is based on Voorhees's query clustering approach [18]. But unlike their method, the importance of the collection is measured by the metadata similarity score generated. The metadata score is simply applied as a factor to each of the individual document similarity scores of each collection's ranked list. The results are then sorted based on this new similarity score and the top N documents returned as results.

5 TREC 11

We have evaluated CARROT II in the context of the TREC 11 Web Track. This track of NIST's annual Text Retrieval Conference focuses on retrieval tasks in web data; in this case, an 18 gigabyte crawl of the .gov domain. The task we chose to attempt was 'topic distillation'. In this task, the goal is to find the best resource page for a given query. Such a page may not necessarily be the page which best matches the content of the query. Other factors, such as links to other relevant pages, may also play a role.

For this task, we used a CARROT II system of 65 agents for indexing and retrieval. Table 1 shows the average document and index sizes. A total of five runs were submitted. The experiments were conducted in two phases. In the first phase the sub-collections were indexed and 1000 results for the 50 topics query generated, under normal CARROT II operation. Two different approaches to query dispersion were used, resulting in two 'base' runs. In the first, queries received were forwarded to all agents in the system. In the second, queries were forwarded to only the ten best agents for that query, based on sub-collection metadata. The results of these base runs were then used as initial seeds and used along with the link topology information provided with the collection. A description of the link topology usage is provided in below:

5.1 Link Topology

The link topology information was used in the following way:

1. Using the link topology file provided by TREC, for each document, a list of inlinks (documents which point to this document) and outlinks (documents which are pointed to by this document) was compiled. The nodes of the file are regarded as nodes of a supergraph G.
2. The 1000 top-ranked documents (seeds), generated from the base runs, are expanded on their inlinks, and a subgraph V is obtained.
3. A directed graph DG on V from G is created.
4. The similarity values are propagated from the seeds on the DG, using one of the 3 formulae (for different runs):

```
(a) Constant c = 1.1
    if (at least half of the children have greater similarity than itself)
        sim = c * avg(best-children)
    else
        sim = avg(sim, avg (worst-children))
(b) Constant c = 1.05
    if (at least half of the children have greater similarity than itself)
        sim = $c^{(number-of-good-children)} *$ avg(best-children).
        sim = min(1+ sim(best-child))/2 , sim)
    else
        sim = avg(sim, avg (worst-children))
(c) Constant c = 1.0
    if (at least quarter of the children have greater similarity than itself)
        sim = $c^{(number-of-good-children}) *$ avg(best-children)
    else
        sim = min(1+ sim(best-child))/2 , sim).
        sim = avg(sim, avg (worst-children))
```

5. The 1000 nodes in DG with the largest sim-value are returned.

Collection	.gov
Average Number of Documents per Agent	19000
Average Document Collection Size	275MB
Average Document Size	6.3KB
Average Index Size per Agent	93MB
Average Metadata Size per Agent	899MB

Table 1. .gov TREC Web Track Collection Statistics

Mean Average Precision	Run1	Run3
At 5 Documents	0.15	0.08
At 10 Documents	0.11	0.05
At 100 Documents	0.03	0.02

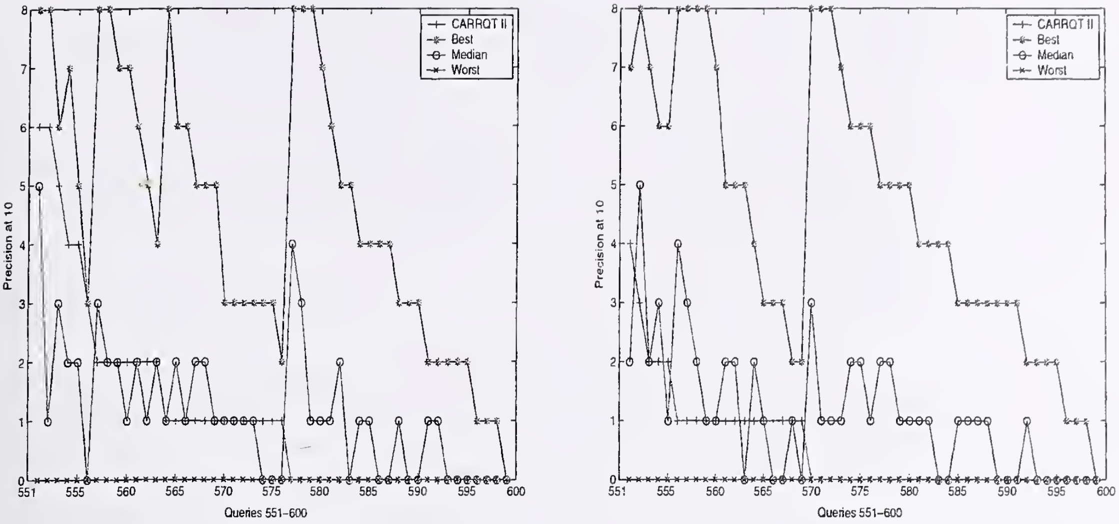
Table 2. *CARROT II* Performance on Topic Distillation Task

5.2 Runs

A description of the five runs submitted is given below:

1. RUN1 (CARROT2A): Up to 65 collections were queried and results fused.
2. RUN2 (CARROT2C): Based on results from RUN1, use link structure and formula a.
3. RUN3 (CARROT2B): Up to 10 collections were queried and results merged.
4. RUN4 (CARROT2D): Based on results from RUN1, use link structure and formula b.
5. RUN5 (CARROT2E): Based on results from RUN1, use link structure and formula c.

Table 2 shows the precision values (provided by TREC) of RUN1 and RUN3 for the 50 queries 551 to 600. The exception was query 582, for which precision values were not provided. We also compared RUN1 and RUN3 precision values with the best, median and worst precision values for each query (see Figure 1). The graph was made by sorting the CARROT II values and using them as reference for the best, median and worst values (The queries represented on the X-asis were sorted in order of decreasing precision values of the CARROT II system). The comparsion showed that for approximately half the number of queries, the CARROT II retrieval effectiveness was close to median. Thus, the simple cosine similarity metric based distributed retrieval approach was able to perform close to the median without enhancements such as query expansion. We believe that these results are encouraging for further improvements in database selection and results fusion techniques for improvements in effectiveness. However, in general, advanced retrieval tasks such as topic distillation on large amounts data maybe outside the scope of purely statistical retrieval techniques. The trial runs involving usage of link structure information did not produce noticably different results from those of RUNS 1 and 3.



(RUN1)

(RUN3)

Fig. 1. Comparison of Precsion Results for RUN1 and RUN3

6 Conclusions

We presented an initial prototype of a Distributed Information Retrieval system on an agent-based architecture. We have built a system that demonstrates that a distributed information retrieval approach may perform at a level comparable to or slightly less than, that of a centralized system with a corpus of the same size. Our approach to topic distilation was based on using the cosine similarity metric with the data divided amongst agents. The observations derived from our participation at the Web Track are:

1. There is a scope of improvement in terms of using just term statistic based cosine similarity metrics, augmented with link topology information and,
2. The potential benefits of distributed retrieval in open environments like the web does make a case further research.

References

1. J. A. Aslam and M. Montague. Models for metasearch. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 276–284, 2001.
2. C. M. Bowman, P. B. Danzig, D. R. Hardy, U. Manber, and M. F. Schwartz. The Harvest information discovery and access system. *Computer Networks and ISDN Systems*, 28(1–2):119–125, 1995.

3. J. Callan. *Advances in Information Retrieval*, chapter 6: Distributed Information Retrieval, pages 127–150. Kluwer Academic Publishers, 2000.
4. J. P. Callan, Z. Lu, and W. B. Croft. Searching distributed collections with inference networks. In E. A. Fox, P. Ingwersen, and R. Fidel, editors. *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* pages 21–28, Seattle, Washington. 1995. ACM Press.
5. R. S. Cost, T. Finin, Y. Labrou, X. Luan, Y. Peng, I. Soboroff, J. Mayfield, and A. Boughannam. Jackal: A Java-based tool for agent development. In J. Baxter and C. Brian Logan. editors. *Working Notes of the Workshop on Tools for Developing Agents, AAAI '98.* number WS-98-10 in AAAI Technical Reports. pages 73–82, Minneapolis, Minnesota, July 1998. AAAI. AAAI Press.
6. R. S. Cost, S. Kallurkar. H. Majithia. C. Nicholas. and Y. Shi. Integrating Distributed Information Sources with CARROT II. In *Proceedings of the Sixth International Workshop on Cooperative Information Agents, CIA 02*. September 2002.
7. G. Crowder and C. Nicholas. Resource selection in CAFE: An architecture for network information retrieval. In *Proceedings of the Network Information Retrieval Workshop, of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval 96*. August 1996.
8. G. Crowder and C. Nicholas. Metadata for distributed text retrieval. In *Proceedings of the Network Information Retrieval Workshop, SIGIR 97*, 1997.
9. T. Finin, Y. Labrou. and J. Mayfield. KQML as an agent communication language. In J. Bradshaw, editor, *Software Agents*. MIT Press. 1997.
10. J. C. French. A. L. Powell. J.' P. Callan. C. L. Viles, T. Emmitt. K. J. Prey. and Y. Mou. Comparing the performance of database selection algorithms. In *Proceedings of the 22th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. pages 238–245. 1999.
11. N. Gibbins and W. Hall Scalability issues for query routing service discovery. In *Second Workshop on Infrastructure for Agents, MAS and Scalable MAS at the Fourth International Conference on Autonomous Agents*. pages 209–217. 2001.
12. L. Gravano and H. Garcia-Molina. Generalizing gloss to vector-space databases and broker hierarchies. In *Proceedings of the 21st VLDB Conference*, Zurich, Switzerland, 1995.
13. A. E. Howe and D. Dreilinger. SAVVYSEARCH: A metasearch engine that learns which search engines to query. *AI Magazine*. 18(2):19–25. 1997.
14. L. Liu. Query Routing in Large Scale Digital Library Systems. *ICDE, IEEE Press*, 1997.
15. C. Pearce and C. Nicholas. TELLTALE: Experiments in a dynamic hypertext environment for degraded and multilingual data. *Journal of the American Society for Information Science*. June 1994.
16. A. L. Powell. J. C. French. J. Callan. M. Connell, and C. L. Viles. The impact of database selection on distributed searching. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–239. 2000.
17. G. Salton. C. Yang. and A. Wong. A vector space model for automatic indexing. *Communication of the ACM*, pages 613–620. 1975.
18. E. M. Voorhees, N. K. Gupta. and B. Johnson-Laird. Learning collection fusion strategies. In *SIGIR*, pages 172–179. 1995.
19. I. H. Witten. A. Moffat. and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Van Nostrand Reinhold. 1994.

CLIR Experiments at Maryland for TREC-2002: Evidence combination for Arabic-English retrieval

Kareem Darwish and Douglas W. Oard[1]
Institute for Advanced Computer Studies
University of Maryland, College Park, MD 20742
{kareem,oard}@glue.umd.edu

Abstract

The focus of the experiments reported in this paper was techniques for combining evidence for cross-language retrieval, searching Arabic documents using English queries. Evidence from multiple sources of translation knowledge was combined to estimate translation probabilities, and four techniques for estimating query-language term weights from document-language evidence were tried. A new technique that exploits translation probability information was found to outperform a comparable technique in which that information was not used. Comparative results for three variants of Arabic "light" stemming are also presented. A simple variant of an existing stemming algorithm was found to result in significantly better retrieval effectiveness.

1 Introduction

Statistical techniques for processing natural language are imperfect, but reliance on multiple sources of evidence can help to mitigate the limitations of any one technique. In this paper, we leverage the combination of evidence in two ways: (1) to estimate translation probabilities from multiple sources of translation knowledge, and (2) to estimate weights for a query term weights based on the statistics of multiple document-language terms. Five translation resources of three types (machine translation lexicons, a printed bilingual dictionary that had been manually rekeyed, and translation probabilities derived from parallel corpora) were used as a basis for estimating translation probabilities. Four ways to estimate query term weights were tried (translation-based indexing, Pirkola's structured query method, and two newly developed variants of structured queries). The main task in the TREC-2002 CLIR track was retrieval of Arabic documents using English queries, so we also made some refinements to our Arabic text processing techniques. Most importantly, we compared three approaches to "light" (i.e., one-level rule-based truncation) stemming. Light stemming was found to perform well for retrieval of character-coded Arabic text in TREC-2001 [1, 2], so we limited our experiments to that technique. We begin with a description of the techniques that we tried in the next section. Section 3 then presents our results, and section 4 concludes the paper.

2 Methodology

In this section, the approaches mentioned in the introduction will be described in detail.

2.1 Translation Resources

Using multiple sources of evidence to guide the translation process can help in two important ways: (1) No single source provides a comprehensive set of translations, and (2) No single source provides an accurate indication of translation probabilities. Drawing translation knowledge from diverse sources therefore offers

[1] College of Information Studies and Institute for Advance Computer Studies.

an interesting potential to improve CLIR effectiveness. We had five translation resources of three types available:

- Two bilingual term lists that were constructed using two Web-based machine translation systems (Tarjim and Al-Misbar [3][4]). In each case, we submitted sets of isolated English words found in a 200 MB collection of Los Angeles Times news stories for translation from English into Arabic [5]. Each system returned at most one translation for each submitted word. Together, the two term lists covered about 15% of the unique Arabic stems in the AFP collection (measured by using light stemming on both the term list and the collection).
- The Salmone Arabic-to-English dictionary, which was made available for use in the TREC-CLIR track by Tufts University. No translation preference information is provided in this dictionary, but it does include rich markup describing morphology and part-of-speech information. We preprocessed the dictionary without using any of that additional information, thereby creating a bilingual term list. The coverage of the resulting term list, measured in the same way, was about 7% of the unique Arabic stems in the AFP collection.
- Two translation probability tables, one for English-to-Arabic and one for Arabic-to-English. These tables were constructed from tables provided by BBN, which were in turn constructed from a large collection of aligned English and Arabic United Nations documents using the Giza++ implementation of IBM's model 1 statistical machine translation design. The coverage of the of the Arabic-to-English table, measured in the same way, was 29% of the unique Arabic stems in the AFP collection.

We combined the evidence from these sources in the following manner:

1. We selected a direction (English-to-Arabic or Arabic-to-English), and then inverted any resources that were originally provided in the other direction. For the translation probability table, we retained the probabilities for each translation pair and then renormalized the inverted table so that the values of the "probabilities" for each source-language term summed to one. This process likely introduced some error, since probabilities for rare events may not have been accurately estimated.
2. A uniform distribution was used to assign probabilities to the translations obtained from machine translation systems and the Salmone dictionary. Tarjim and Al-Misbar returned at most one translation for an English word, but two English words might share a common translation. When n alternatives were known from a single source, each was assigned a probability of $1/n$.
3. For translation from English to Arabic, the resulting translation probabilities were then combined across sources by summing the probabilities for a given Arabic translation across the sources in which it appeared and then dividing by the number of sources in which the English term had appeared. For example, if Tarjim, Al-Misbar and Salmone contained the English term, with Tarjim containing some specific translation with probability 1.0, Al-Misbar lacking that translation (i.e., assigning it a probability of 0.0), and Salmone assigning it a probability of 0.5 (because two translations were known), then the resulting combined probability would be $1/3 + 0/3 + 0.5/3 = 0.5$. Translation probabilities for Arabic-to-English were computed in the same manner, but with the role of each language reversed.

The resulting translation resource contained what appeared to us to be reasonable estimates of translation probabilities, and covered 36% of the unique Arabic stems in the AFP collection.

2.2 Stemming

Three Arabic light stemmers were tested. The first was Al-Stem, the stemmer provided by the organizers for the standard resource run, which was developed by the author at Maryland and modified by Leah Larkey from University of Massachusetts (U Mass). The second was a light stemmer described in Larkey's SIGIR 2002 paper, which we will refer to as the U Mass stemmer [2]. The third was a modified version of the U Mass stemmer in which two additional prefixes identified through failure analysis using the TREC-2001 topics, were removed. Table 2 lists the prefixes and suffixes removed by each stemmer. Of the three, Al-Stem was the most aggressive stemmer.

Stemmer	Prefixes	Suffixes
Al-Stem	wAl, fAl, bAl, bt, yt, lt, mt, wt, st, nt, bm, lm, wm, km, fm, Al, ll, wy, ly, sy, fy, wA, fA, lA, and bA. (والـ، فالـ، بالـ، بتـ، يتـ، لتـ، متـ، وتـ، ستـ، نتـ، بمـ، لمـ، ومـ، كمـ، فمـ، الـ، للـ، وي، لي، في، وا، فا، لا، با)	At, wA, wn, wh, An, ty, th, tm, km, hm, hn, hA, yp, tk, nA, yn, yh, p, h, y, A. (ـات، وا، ون، وه، ـان، تي، ته، تم، كم، هم، هن، ها، ية، تك، نا، ين، يه، ـة، ـه، ي، ا)
Umass Stemmer	Al, wAl, bAl, kAl, fAl, and w (الـ، والـ، بالـ، كالـ، فالـ، و)	hA, An, At, wn, yn, yh, yp, p, h, and y (ـي، ـه، ـة، ية، يه، ين، ون، ـات، ـان، ها)
Modified UMass Stemmer	-- Identical to U Mass, plus ll, and wll (للـ، وللـ)	-- Identical to U Mass --

Table 2: Prefixes and suffixes removed by each light stemmer.

2.3 Combination of Evidence for Alternate Translations

One of the key challenges in CLIR is what to do when more than one possible translation is known. One principled solution to this problem is Pirkola's structured query method [6]. Pirkola used InQuery's synonym operator to estimate the weight of each query term based on document language evidence as follows:

$$TF_j(Q_i) = \sum_{\{k \mid D_k \in T(Q_i)\}} TF_j(D_k) \qquad (1)$$

$$DF(Q_i) = \left| \bigcup_{\{k \mid D_k \in T(Q_i)\}} \{d \mid D_k \in d\} \right| \qquad (2)$$

Where Q_i is a query term, D_k is a document term, $TF_j(Q_i)$ is the term frequency of Q_i in document j, $DF(Q_i)$ is the number of documents that contain Q_i, d is a document, and $T_j(D_k)$ is the set of known translations for the term D_k. The key insight here is that translation is viewed as a process akin to stemming, in which several document-language terms might be treated as if it were the same query language term (in stemming, many morphological variants are treated as if they were the same stem). One limitation of Pirkola's structured query method is that it makes no use of translation probabilities—every possible translation is treated as being equally likely. Consider a case in which one query term has two translations, one of which is highly probable, but which typically has a low term frequency when it appears, while the other is improbable but typically has high term frequencies. Because the improbable translation is so common, documents containing that translation would likely be retrieved ahead of documents that contain the more likely translation. This seems undesirable. Moreover, consider a second case in which an extremely improbable translation appears in many documents, with an almost certain translation appears in only a few. Intuitively, it would seem that we should ignore the extremely improbably translation and assign this term a low document frequency (and thus give it added importance in any retrieval system that relies on inverse document frequency as a measure of term importance). But Pirkola's structured query method does just the opposite.

In this paper, we propose two ways of incorporating translation probability information. The simplest approach is to retain the same formulae, but to suppress the contribution of unlikely translations. We implemented this by starting with the most likely translation and adding additional translations in order of decreasing probability until the cumulative probability of the selected translations reached a preset threshold that was determined through experimentation using the TREC-2001 CLIR collection. As an alternative, we also explored three ways of incorporating translation probabilities directly into the formulae:

1. The weighted DF method (WDF): In this method, translation probability estimates are used in the calculation of document frequency, but not term frequency. Formally, $TF(Q_i)$ is computed as in equation (1), and $DF(Q_i)$ is computed as follows:

$$DF(Q_i) = \sum_{\{k \mid D_k \in T(Q_i)\}} \left[DF_j(D_k) \times wt(D_k) \right] \qquad (3)$$

Where Q_i is a query term, D_k is a document term, $DF(Q_i)$ is the number of documents that contain Q_i, $wt(D_k)$ is the probability estimate for D_k, and $T_j(D_k)$ is the set of translations for the term D_k. This method addresses the case in which an improbable translation has a high document frequency. Note that the union operator has been replaced by the sum operator, so equations (2) and (3) will not necessarily produce the same value, even in the case of equiprobable translations.

2. The weighted TF method (WTF): In this method, translation probability estimates are used in the calculation of term frequency, but not document frequency. Formally, $DF(Q_i)$ is computed as in equation (2), and $TF(Q_i)$ is computed as follows:

$$TF_j(Q_i) = \sum_{\{k|D_k \in T(Q_i)\}} \left[TF_j(D_k) \times wt(D_k) \right] \qquad (4)$$

   Where Q_i is a query term, D_k is a document term, $TF_j(Q_i)$ is the term frequency of Q_i in document j, and $T_j(D_k)$ is the set of different translations for the term D_k. This method addresses the case in which an improbable translation typically has high term frequencies.
3. The weighted TF/DF method (WTF/DF): In this method, translation probability estimates are used in the calculation of both term frequency and document frequency. Equation (3) is used to compute $DF(Q_i)$, with equation (4) used to compute $TF_j(Q_i)$. This addresses both potential concerns.

The same approach to preselection of the most likely translation probabilities that we used with Pirkola's structured queries can also be used with any of these methods.

2.4 Balanced Translation-Based Indexing

The formulae for Pirkola's method do not depend on any information that can only be computed at query time, so an indexing time implementation is possible. Oard and Ertunc built such an implementation, which they called translation-based indexing. That approach is equivalent to unbalanced document translation, in which every English translation for each Arabic term is indexed once. Unbalanced translation is, however, known to underemphasize the contribution of highly specific terms (which typically have very few possible translations) and to overemphasize the contribution of common terms (which often have many possible translations). The obvious alternative is to balance the translation-based indexing process. To achieve this, we replaced each Arabic term with the five most likely English translations. For terms with fewer than 5 known translations, the most probable translations were replicated in a round-robin fashion until a total of five was reached. We chose the value 5 based on post-hoc experiments with the 25 TREC 2001 topics (we tried 1, 3, 5, 7, and 10).

2.5 Document and Query Expansion

Any individual document will likely include only a fraction of the words that might be used to describe a topic, so some form of expansion to the representation of a document might help retrieval. We therefore prepared a contrastive condition in which the representation of each document was expanded as follows:

1. We identified the 20 most descriptive terms in each document, by dividing the frequency with which each term appeared in the document by the number of documents in which that term was found.
2. We then formed a query with one instance of each of those 20 terms and used that query as a basis for ranking the documents in the AFP collection using the InQuery text retrieval system from the University of Massachusetts (the document being expanded was often in this set). We used the Al-Stem stemmer to normalize the representation of Arabic terms in InQuery, both for query and for document processing.
3. We combined the 10 top-ranked documents into a single mega-document and then chose the 20 most descriptive terms in that mega-document, using the same measure of term importance as above. The resulting set of 20 terms was then added to the representation of document that was being expanded.

One can imagine many variants of this approach, including alternative parameter settings, alternative term importance measures, and alternative ways of combining evidence from the top-ranked documents. Because document expansion was not the principal focus of our experiments, we tried only this one implementation for TREC-2002.

Because queries are relatively brief, they are even more likely to contain only a small subset of the words that might be used to express the concepts that are important for the topic that they represent. We therefore performed pre-translation query expansion using a similar blind relevance feedback process. We used Associated Press articles from 1994-1998 from the North American News Text Corpus (Supplement) and the orld Stream English Collection from the Linguistic Data Consortium for this purpose Because InQuery contains built-in provisions for query expansion, we used InQuery (with the kstem English stemmer) to search these collections. We configured InQuery to choose the most discriminating terms from the top 10 returned documents for each query.

2.6 Implementation Details

To ease work in Arabic, Arabic letters were transliterated to Roman letters. Table 1 shows the mappings between the Arabic letters and their transliterated representations. In preprocessing the text, all diacritics and kashidas were removed, and letter normalization was employed to normalize the letters ya (ي) and alef maqsoura (ى) to ya (ي) and all the variants of alef (ا) and hamza (ء), namely alef (ا), alef hamza (إ ،أ), alef maad (آ), hamza (ء), waw hamza (ؤ), and ya hamza (ئ), to alef (ا).

أ، ا، آ، إ	A	ئ، ؤ، ء	A	ب	b
ت	t	ث	v	ج	j
ح	H	خ	x	د	d
ذ	O	ر	r	ز	z
س	s	ش	P	ص	S
ض	D	ط	T	ظ	Z
ع	E	غ	g	ف	f
ق	q	ك	k	ل	l
م	m	ن	n	ه	h
و	w	ي، ى	y	ة	p

Table 1: English transliteration of Arabic characters

As for a stop-word list, we used the list that is distributed with Sebawai, which includes 130 particles and pronouns [7]. Sebawai is a publicly available Arabic morphological analyzer. We used InQuery for our official monolingual runs and for one of our official cross-language runs. For the remaining monolingual runs and for our post-hoc experiments, we used PSE (an IR system developed at Maryland that uses OKAPI BM-25 weights) because we were not able to perform the necessary changes to the term weight computation using InQuery. Our monolingual and cross-language results should therefore not be considered to be strictly comparable.

3 Results

We submitted five official runs, and we have used the relevance judgments provided by NIST to perform an additional 31 post-hoc runs. In this section we present those results along with a preliminary discussion that we intend to extend in our final paper.

3.1 Arabic Monolingual Runs

We submitted one official automatic monolingual run, one official manual monolingual run, and we performed three post-hoc automatic monolingual runs. In the official automatic run, queries were formed using every word in the title and description fields of the topic description (stopwords were removed, but no automatic stop structure removal was performed). For the manual run, the title, description, and narrative fields were used, additional query terms were manually added, and stop structure and information about what would cause a document to be judged as not relevant was removed manually. The principal goal of the manual run was to enrich the relevance pools. In both cases, Al-stem was used to stem the documents and the queries, and document expansion was used as described above. Table 3 shows the results (results on TREC-2001 data are post-hoc).

The three post-hoc runs were done using PSE rather than InQuery, with a goal of comparing three variants of light stemming. Document expansion was not used in those experiments. Again, Table 3 shows the results. The Al-stem and the U Mass stemmers were found to be statistically indistinguishable by a paired two-tailed t-test (for $p<0.05$). The modified U Mass stemmer did achieve mean average precision results that were statistically better than the two other stemmers over the full 75-topic set, but the advantage over the unmodified U Mass stemmer was not seen when the 50 TREC 2002 topics used alone. Because the suitability of the TREC-2001 collection for post-hoc experiments is open to some question, we conclude that no reliable differences between the three stemmers were detected by our experiments.

Run	Mean Average Precision	
Official Runs (with document expansion)	TREC 2002 Topics (50 topics)	TREC 2001 & 2002 Topics (75 topics)
Automatic Monolingual	0.289	0.314
Manual Monolingual	0.302	0.322
Post-Hoc Runs (no document expansion)	TREC 2002 Topics (50 topics)	TREC 2001 & 2002 Topics (75 topics)
Al-Stem stemmer	0.286	0.316
U Mass stemmer	0.295	0.321
Modified U Mass stemmer	0.301	0.331

Table 3: The Arabic Monolingual Results

3.2 English-Arabic CLIR Runs

We performed three official automatic CLIR runs and 29 post-hoc automatic CLIR runs. In each case, we formed title+description queries in the same manner as for the automatic monolingual run. Pre-translation query expansion was done using blind relevance feedback on Associated Press articles from 1994-1998 using InQuery. The articles were part of the North American News Text Corpus (Supplement) and AP World Stream English Collection from the Linguistic Data Consortium [8]. For the official CLIR runs we tried these following configurations:

- **Pirkola's Method.** We used InQuery, pre-translation query expansion, document expansion, and the thresholded version of Pirkola's method (with a threshold of 0.3, established using post-hoc experiments with the TREC 2001 topics).
- **Balanced Translation-Based Indexing (TBI).** We used InQuery, document expansion, and balanced translation-based indexing.
- **Weighted TF.** We used PSE, pre-translation query expansion, document expansion, and the Weighted TF method (with a threshold of 0.35, again tuned using the TREC-2001 topics).

For the post-hoc experiments, we used PSE, pre-translation query expansion, one of four methods (Pirkola's method, Weighted TF, Weighted DF, or Weighted TF/DF), and a probability threshold that was varied between 0.1 and 0.7 in increments of 0.1. Post-hoc CLIR results are reported on all 75 topics from TREC 2001 and TREC 2002. We also conducted one additional post-hoc experiment in which we combined the results from our official Weighted TF and translation-based indexing runs in a manner similar to that suggested by [9]. For that experiment, we renormalized the PSE score for the Weighted TF run to be between 0.4 and 0.5 (a range similar to that seen in InQuery), averaged that normalized score with the translation-based indexing score from PSE, and resorted the ranked list based on that averaged score. The result was statistically better than the Weighted TF run, but not statistically distinguishable from the translation-based indexing run. Table 4 shows the results for the official runs and for the one post-hoc experiment that was based on the two official runs. Table 5 and Figure 1 together show the other post-hoc results.

Run	Mean Average Precision	
Official Runs (Expanded documents)	TREC 2002 Topics (50 topics)	TREC 2001 & 2002 Topics (75 topics)
Pirkola's Method	0.202	0.252
Balanced TBI	0.274	0.279
Weighted TF	0.247	0.279
Balanced TBI + Weighted TF	0.289	0.307

Table 4: CLIR results for official runs (expanded documents).

	Cumulative Probability Threshold						
Method	0.1	0.2	0.3	0.4	0.5	0.6	0.7
Pirkola's Method	0.215	0.212	0.230*	0.230	0.207	0.186	0.134
Weighted TF	0.220	0.226	0.241	0.247*	0.241	0.230	0.192
Weighted DF	0.211	0.196	0.198	0.180	0.147	0.126	0.091
Weighted TF/DF	0.220	0.222	0.235	0.234	0.236	0.240	0.245

Table 5: Post-hoc CLIR results (Mean Average Precision, 75 topics, no document expansion.

The Weighted TF method and translation-based indexing were both statistically significantly better than Pirkola's method (on the 75 topic set), but statistically indistinguishable from each other. The retrieval effectiveness of Weighted TF/DF was close to the best at every cumulative probability threshold, and it was the only one of the four techniques that we tried that did not exhibit a decrease in effectiveness at high thresholds. It therefore seems to be a good candidate for further study, and an appropriate choice if a method

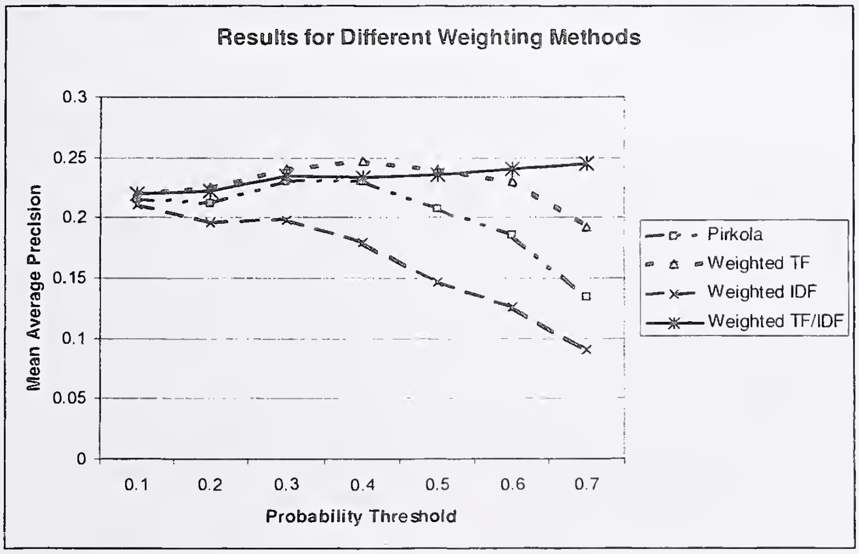


Figure 1: Sensitivity to cumulative probability threshold.

must be selected without access to a representative collection on which to tune the probability threshold. We do not yet know whether document expansion was helpful because the runs with and without document expansion were done using different retrieval systems.

4 Conclusion

We have presented two basic ideas for using a combination of evidence to improve cross-language retrieval effectiveness, demonstrated an ability to produce useful translation probability estimates from multiple sources, and extended Pirkola's structured query method in ways that exploit that information. Our results suggest that further work on the Weighted TF/DF method would be well justified, and that further work on document expansion is needed before the utility of that technique in this context can be judged. We look forward to the discussions that we will have at the conference, and to the opportunity to continue out exploration of these questions through additional post-hoc experiments and analysis.

Acknowledgments

This work was supported in part by DARPA Cooperative Agreement N660010028910.

References

[1] Aljlayl, M., S. Beitzel, E. Jensen, A. Chowdhury, D. Holmes, M. Lee, D. Grossman, and O. Frieder, "IIT at TREC-10," TREC: 265-274, 2001.

[2] Larkey, L., L. Ballesteros, and M. Connell, "Improving Stemming for Arabic Information Retrieval: Light Stemming and Co-occurrence Analysis," SIGIR 2002: 275-282, 2002.

[3] tarjim.ajeeb.com, Sakhr Technologies, Cairo, Egypt www.sakhr.com

[4] www.almisbar.com, ATA Software Technology Limited, North Brentford Middlesex, UK

[5] NIST, Text Research Collection Volume 5, April 1997.

[6] Oard, W. and F. Ertunc: Translation-Based Indexing for Cross-Language Retrieval. ECIR 2002: 324-333, 2002.

[7] Darwish, K., "Building a Shallow Morphological Analyzer in One Day," ACL Workshop on Computational Approaches to Semitic Languages: 47-54, 2002.

[8] MacIntyre, Robert, "North American News Text Supplement", LDC98T30, LDC, 1998.

[9] McCarley, S., "Should we Translate the Documents or the Queries in Cross-Language Information Retrieval," ACL, 1999.

Video Indexing and Retrieval at UMD

Christian Wolf
Laboratoire RFV
INSA de Lyon
69621 Villeurbanne cedex, France
wolf@rfv.insa-lyon.fr

David Doermann
Language and Media Processing Lab.
University of Maryland
College Park, MD 20742-3275, USA
doermann@cfar.umd.edu

Mika Rautiainen
MediaTeam Oulu
University of Oulu
P.O.BOX 4500, Finland
mika.rautiainen@ee.oulu.fi

Abstract

Our team from the University of Maryland and INSA de Lyon participated in the feature extraction evaluation with overlay text features and in the search evaluation with a query retrieval and browsing system. For search we developed a weighted query mechanism by integrating 1) text (OCR and speech recognition) content using full text and n-grams through the MG system, 2) color correlogram indexing of image and video shots reported last year in TREC, and 3) ranked versions of the extracted binary features. A command line version of the interface allows users to formulate simple queries, store them and use weighted combinations of the simple queries to generate compound queries.

One novel component of our interactive approach is the ability for the users to formulate dynamic queries previously developed for database applications at Maryland. The interactive interface treats each video clip as visual object in a multi-dimensional space, and each "feature" of that clip is mapped to one dimension. The user can visualize any two dimensions by placing any two features on the horizontal and vertical axis with additional dimensions visualized by adding attributes to each object.

1 Introduction

For the search evaluation we ran experiments for both run types using two different tools: a command line based query tool which allows us to query a precompiled database using text and speech (keywords), color, and binary feature results and a data browsing tool which represents each shot as a point in a space defined by the available features.

This paper is outlined as follows: the text detector used for feature extraction has been described previously in [12]. The experiments specific to the TREC competition are described in Section 2. The features used for the search task and the techniques we developed to query them are given in Section 3. Experiments are presented in Section 4, and Section 5 provides conclusions and describes the future directions of our research.

2 The feature extraction task: detection of overlay text

Our team participated in the feature extraction subtask with an overlay text detector developed at INSA de Lyon [12]. Text is detected on a frame by frame basis, integrated into a single image per occurrence, enhanced and binarized before being passed to a commercial OCR. For the TREC competition we used OCR software from Scansoft, which does not perform as well on our data as the Finereader software we used in [12], but it does come with a programable API which makes it easier to integrate it into our system. This is imperative given the amount of data we needed during the competition.

In order to increase the precision of the detector, we developed a classifier which takes the OCR output from each detected text box and classifies it either as a positive text result or a false alarm.
Table 2 shows OCR output examples for text and non-text. At first glance it seems to be straightforward for a human to judge between "meaningful" text and junk strings. However, we did not want to use a dictionary based approach for two reasons:

Non text examples		Text examples	
a yen Pu s1c~	i ~~~t~ ..~f: a1 t.	TONY RIYERA	Art Direction
v\~~~	ad	EUGENE PODDANY	URANIUM
.~a~~ 1r.	~! ~ ~._’_‘	EMERY NAWK~N5	92
~_	,(1f~rJ1	GERALD NEYIU	GzOTONS 92
.1	I~rs~-’.’ i~u .r	D i recto r	arrtoms 142
j_ I	lsYf,7	CARL URBAN	234

Table 1: Examples for text and non-text OCR output.

- Screen text often contains words which cannot be found in a dictionary, such as names and identifiers containing digits.
- A part of the OCR output is very noisy (e.g. “arrtoms” instead of “atoms” in the examples).

We therefore used simple features based on the type and distribution of the characters in the string. We manually classified the set of possible characters into the 4 different groups “upper case”, “lower case”, “digits”, “non-text”. The input to the classifier is a feature vector containing the following two features:

$$F_1 = \frac{\text{Number of good characters (upper+lower+digits)}}{\text{Number of characters}} \qquad F_2 = \frac{\text{Number of group changes}}{\text{Number of characters}}$$

We trained a linear classifier with Fisher’s linear discriminant [1] on a set of training strings and ranked all text instances by the classifier output.

3 Search: features and query techniques

For the second task, general search, we used all of the donated features (i.e. the results of the automatic speech recognition (ASR), the binary features, as well as the results of the automatic text recognition (ATR) described in Section 2 and the corresponding transcripts, and the temporal color correlograms already used during TREC 2001 and described in [9]).

In the following subsections we describe the different features and the query techniques necessary to use them in a query system. Finally, Sections 3.4 and 3.5 describe the two tools employed for our experiments.

3.1 Recognized text and speech

Recognized video text and speech transcriptions are treated in the same way by our system. For retrieval we used the Managing Gigabytes system[11] which is freely available[1]. This software is able to retrieve “documents” from one or more “collections”, defined by the user. In our case, documents were the text content obtained for each shot, with the three collections corresponding to different feature sources: Text, Speech and Text+Speech. Managing Gigabytes (MG) allows users to choose between two different query metrics: boolean queries and ranked queries, where the relevance of each shot is determined by the cosine similarity[11]. We allowed users to choose between the boolean query type and the ranked query type, with ranked queries being the default.

MG stems all document and query terms, so for example, looking for the term “produced” will also retrieve shots containing the term “producer” or “producing”. Unfortunately this feature cannot be turned off, so we could not measure it’s influence on the query results in the case of noisy transcripts.

[1] http://www.cs.mu.oz.au/mg

The MG system was developed for large collections of noise free documents, and therefore all hits are computed as exact matches (apart from the stemming technique). The ASR transcripts we received from the feature donors had been post processed with a dictionary and therefore contained only noise free text. The ATR transcripts on the other hand were quite noisy. For example the string "`Nick Chandler`"[2] had been recognized by the OCR in the shot as "`ni ck 16 tia ndler`", so a search for the keyword "chandler" would not retrieve the shot. We included a possibility of performing inexact matches using N-grams. The N-Gram of a string is the set of all sub strings of length $\geq N$. By performing a boolean query and OR-connecting all the substrings, exact matches of these substrings will be retrieved. The minimum length N of the substrings determines the level of "exactness". Lower values of N will retrieve documents with noisier keywords but also more irrelevant documents. We set the N empirically to 5. Applied to the example given above, an inexact query on the keyword "chandler" would result in the following boolean query:

```
chand|handl|andle|ndler|chandl|handle|andler|chandle|handler|chandler
```

which retrieves the desired shot but also a shot containing the transcript "`colleges cattlemen handlers of livestock`", clearly a false alarm.

3.2 Binary features

We used all of the donated binary features except CMU's face detector and MSRA's overlay text detector. These two detectors do not provide confidence information, which is needed for our feature combination technique.

Combining the features can be viewed as the classical problem of combining the results of multiple classifiers which has already received considerate attention in the pattern recognition and in the retrieval community [5, 3, 2, 6]. Training the combining classifier has been reported to considerately improve the performance of the total system in some cases, especially if the base classifiers use different features[2]. In our case, however, training data is not available since we are not allowed to use features from TREC's search test collection. Therefore, using fixed rules to combine the classifiers is the only available option.

In the following discussion, $C_{ij}(x)$ denotes the output of classifier j for class i, given the measurement vector x and $Q_i(x)$ the output of the combining classifier. We did not formulate the rules in a probabilistic way, since we cannot know if the output of the classifiers quantifies the posterior probabilities of the class memberships. The two most well known fixed combination rules are the product rule

$$Q_i(x) = \prod_j C_{ij}(x)$$

and the sum rule

$$Q_i(x) = \sum_j C_{ij}(x)$$

[5, 2]. If the feature representations are statistically independent and the classifiers produce posterior probabilities, then the product rule will produce an optimal classifier in the sense that it quantifies the true likelihood of the measurements given class k. Unfortunately, this is rarely the case [5]. The sum rule is considered to work well in cases where base classifiers have independent noise behavior as errors in one classifier are averaged out by the other classifiers.

We chose the more robust sum rule for this task, since we do not have enough information on the type of classifiers used for the base classifiers and how they have been trained. If one of the classifiers is

[2] The person "Nick Chandler" is not related to the "James H. Chandler" from TREC search topic 75.

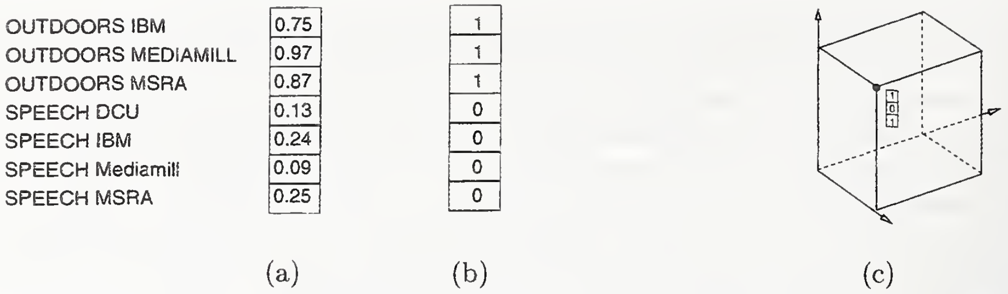


Figure 1: An example shot vector (a) and an example query vector (b) for the query "*outdoors but not speech*". The feature space for a query on 3 features (c). The query vector is a corner point in the hyper cube.

weakly trained, then the product rule will create a combining classifier with very poor results, whereas the sum rule tends to reduce the effect of a poor base classifier. Since we do not want to classify the shot by the combined classifier alone, but are also interested in the output value of the classifier, we divide the output by the number of classifiers, which gives us the average of the classifier outputs. Combining the features this way, it is possible to create filters on the binary features, retrieving only shots with a certain confidence of containing a feature.

On the other hand, a user might prefer to submit a ranked query returning a set of shots "closest to " the desired features. In this case it is more convenient to use a vector space model for the set of classifiers since vector space distance functions can be used to order the shots. The actual space depends on the query specified by the user. The user may wish to retrieve shots with constraints on certain features but may also want to ignore others. As an example, in order to retrieve people walking on a beach, a user might want to return shots containing high confidences on the features *outdoors* and *people* and low confidences on the features *indoors* and *cityscape*, but is likely to ignore the feature *speech.* In our vector space model, each dimension in the vector space corresponds to a donated feature, which is constrained in the query, so the vector space is a sub space of the space spanned by all features. If, for example, the query puts constraints on the features *outdoors* and *speech* and *outdoors* has been donated by 3 teams whereas *speech* has been donated by 4 teams, then the feature space has $N = 7$ dimensions (see Figure 3.2a). For each shot s, the confidence outputs can be placed into a N dimensional feature vector u_s.

The shot vectors are compared with a query vector q, whose elements of the are set to 1 if the corresponding feature is desired by the user and to 0 if the corresponding feature is not (see Figure 3.2b). The space of all possible shot vectors is therefore bounded by an N-dimensional hypercube and the query vector is one of the corners of the hypercube (see Figure 3.2c).

The confidence output of all feature detectors is already scaled to the interval [0−1], but the distribution of the confidence values in the interval may be different for different detectors, as is the case for the two features for *Indoors* donated by IBM and MSRA. We therefore considered the Euclidean distance $D_E(q, u_s) = \{(q-u_s)^T(q-u_s)\}^{-1}$ as well as the Mahalanobis distance $D_M(q, u_s) = (q - u_s)^T \sum^{-1}(q - u_s)$, which scales the distances according to the covariances Σ of the data set. During the experiments, however, the Euclidean distance outperformed the Mahalanobis distance by far. This can be explained with the different distributions of the confidence values of the different detectors.

3.3 Temporal color features

We incorporated a capability for color motion search by the temporal color features developed by the University of Maryland in collaboration with the University of Oulu and presented in [9]. These features model the statistical distribution of the colors in the shot, but, unlike histograms, they also capture the spatial and temporal distribution of the colors. A correlogram is defined as:

$$\gamma_{c_i,c_j}^{(d)} = \Pr_{p_1 \in I_{c_i}^n, p_2 \in I^n} \left(p_2 \in I_{c_j}^n \middle| |p_1 - p_2| = d \right)$$

where I^n are the frames of the shot, p_1 and p_2 are two pixels and c_i and c_j are two colors. In each bin, the feature stores the probability that given any pixel p_1 of color c_i, a pixel p_2 at distance d is of color c_j among the shots frames I^n. For the Video TREC evaluation, the features have been computed as auto correlograms, and therefore $c_i = c_j$, resulting in a two dimensional structure. The distance between two features is calculated using the L_1 norm.

3.4 Querying

A command line interface allows the users to submit queries on the features with the techniques described above. The query results can be stored in 30 different result frames, allowing combinations of the results using different schemes. The following operations are supported by the interface:

- Keyword based queries on text, speech or both; with or without n-grams; and boolean or ranked.
- Ranked color queries.
- Ranked queries on binary features and filters on binary features.
- Combination of query results (AND/OR) including weighted combinations of the ranking of both queries. Assigning each query i a weight α_i, the shots in the combined set are ordered by a measure[3] m_s, which can be calculated from each shot s's ranking $r_{s,i}$ in the query results: $m_s = \sum_i \alpha_i \left(1 - \frac{r_{s,i}-1}{N+1}\right)$ where N is the total amount of shots in the combined set.
- Inspection of the keyframes of queries (thumbnails and full sized image).

3.5 Browsing

In addition to the command line query interface, the data set can be viewed graphically in a browsing tool developed at the University of Maryland. The user has the choice to either viewing the complete data set or to select a set of query results and to view only shots which have been returned by one or more of these queries. For each shot, the following features are available:

- The confidence of the binary features (one value per feature type).
- The text and speech recognition output and it's confidence values.
- The rank of the shot. The ranking information is scaled and normalized to make it intuitively conform to the confidence values of the binary features (*"higher means better"*): $m_s = 1 - \frac{r_{s,i}-1}{N+1}$
- A OR-combined rank-like measure computed on the ranking of the shot in all chosen queries. The weights α_i are chosen by the user when the queries are exported to the browsing tool.

Browsing allows users to view the shots as visual objects on a dynamic two-dimensional grid. The available features can be used in multiple ways to visualize the data set. The user can visualize the shots in two dimensions by placing any two features on the horizontal and vertical axis. Additional dimensions can be visualized by adding attributes to each object. Color, for example, can be used to represent a third feature dimension. Dynamic range sliders are provided for all features (not just the visible ones) so the user can dynamically modify the visibility of each feature and navigate through the collection. Attributes of each object can be configured and displayed as "tips" when the mouse passes near them.

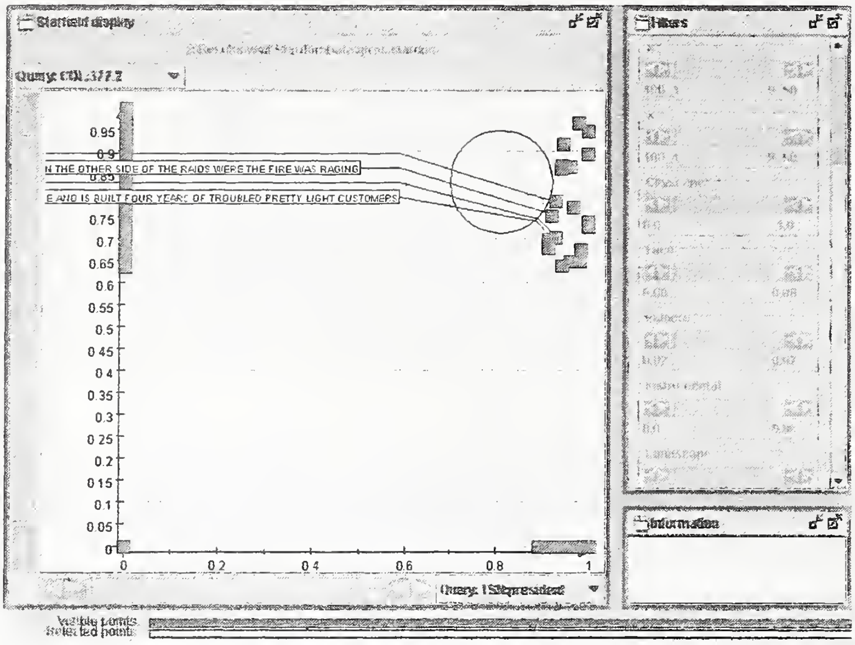


Figure 2: The graphical browsing tool "stardom".

Additionally, by clicking on a visual object or choosing a set of objects, their keyframes can be displayed and a media player can be launched to view the shot.

Figure 3.5 shows an example for a browsing session. The rank of a color query has been assigned to the y-axis and the results of a text/speech query (keyword: "president") has been assigned to the x-axis. All shots inside or touching the red circle are listed in the blue boxed list, for each shot the text/speech recognition results are displayed.

4 Experimental Results

In accordance with the TREC task definitions, we submitted 3 different runs:

Manual run The user adjusts the query parameters in the command line query tool to manually run queries and combines them to a single query. Our users where asked to write down the manual queries on a sheet of paper before actually performing them with the tool in order to avoid potential feedback from the subqueries.

Manual run/no ASR The same queries as above, but without speech recognition.

Interactive run The user is free to look at the results of each intermediate query returned by the command line tool and to adjust parameters and re-run them. Additionally, the query results can be viewed with the browsing tool for a closer inspection of the result sets and the mappings into the feature space. Based on these observations, queries can be re-run in the query tool. Hence, submitted results were queries run and combined in the query tool.

Four different users participated at the experiments. The users had been given dummy topics in order be able to get used to the system and the features. They did not know the real topics before they started working on them.

[3] The measure is not real ranking information since a value may occur several times in the combined set.

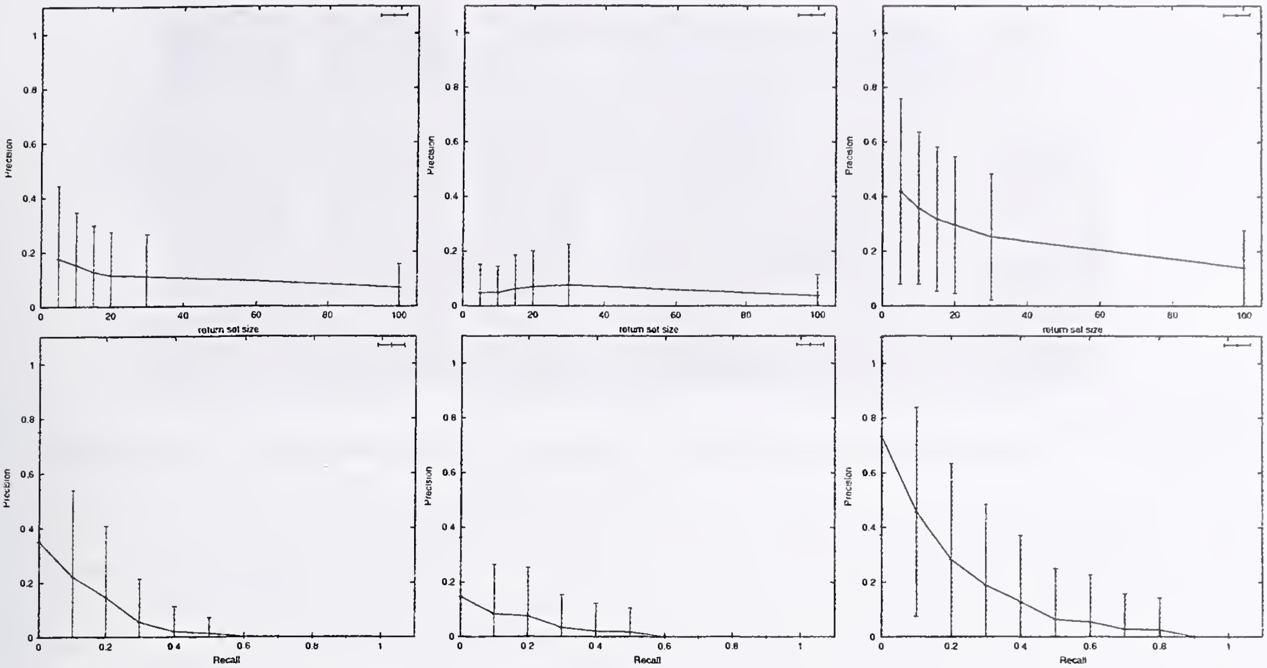


Figure 3: Precision at different result sizes averaged across the topics (top), and precision at different levels of recall averaged across the topics (bottom). From left to right: the manual run, the manual run without speech recognition and the interactive run

4.1 Search results

Figure 3 shows the precision curves vs. result size and recall, respectively, in graphical form. Recently, a new performance graph for the evaluation of content based image retrieval systems has been proposed[4]. It's advantage is the inclusion of generality information, i.e. information about the clustering of the relevant information in the database. This is essential if different systems operating on different databases are compared. Unfortunately, the measure cannot be computed for the TREC experiments, because it requires access to the precision at arbitrary sizes of the result set, information which is currently provided by NIST. Since all teams used the same database, it is possible to compare the different methods with the classic precision/recall graphs in order to determine which method is the best. However, a quantitative evaluation of the differences between the methods also makes the inclusion of the database generality necessary.

The interactive system performed best for query topics *76-James Chandler* (54% precision at 80% recall) and *97-living cells* because of the color features, and for *89-Butterfly* because of the color features and the ASR transcripts. Topics with medium sucess were *80-Musicians*, where a combination of color and ASR queries was used, and *77-George Washington*, *78-Abraham Lincoln*, *90-mountain* and *99-rockets*, which had been tackled with ASR only. For these topics, users performed queries with different keywords which were more or less related to the topic, and filterted the output with the binary features. For topic *86-Cities* a recall of 20% was achieved by a sole usage of color features. The user fed the results of the color queries back to new color queries and combined these results.

During the experiments we had signficant difficulty with the topics *79-Beach*, *87-Oil fields*, *88-Map*, *96-flag*, *98-locomotive approaching*. For four topics our experimenters could not find a single relevant shot: *75-Eddie Rickenbacker*, *81-Football players*, *85-Square arch*, *91-Parrots*. These are cases where the ASR transcripts did not return any relevant shots and the color features were not useful. Searching for

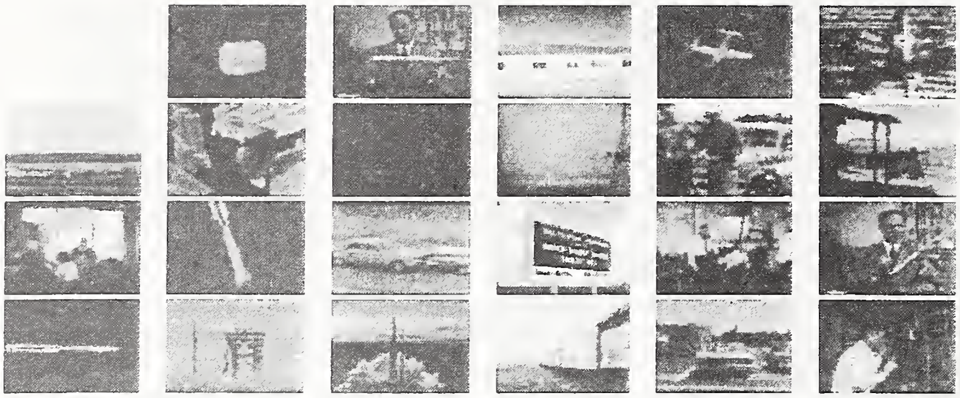


Figure 4: The results of a query on the speech transcripts using the keywords "rockets missile".

Eddie Rickenbacker, even the usage of the context taken from the request statement (e.g. the knowledge that he is the CEO of an airline) did not help to find a shot. The color features were particularly powerless in the topics *football*, *Beach* and *locomotive*, which are specified where a connection between the semantic concept and colors is very hard. A feature detecting drawings and cartoons could have been useful for the map topic. In some cases the users missed a possibility to explicitly specify colors or color ranges (e.g. for the topics *flag* and *parrots*).

4.2 The impact of ASR

Naturally, as the speech recognition transcripts are very close to the semantic meaning of the video, they have a significant impact on the search results. Nevertheless, the quality of keyword based queries on the transcripts highly depends on the topic. In general, the result sets of speech queries are very heterogeneous and need to be filtered, for example, by the results of binary filters.

Consider for example the results on the keywords "rocket, missile", shown in Figure 4. Adding additional keywords (e.g. "taking off") could increase the chances of retrieving rockets actually taking off as opposed to rockets on desks etc., but it also "thins" the semantic meaning of the query, in this case decreasing the probability to find shots about rockets or missiles in the result set. Another possibility is filtering the results of speech queries by the results of binary queries, a method applied frequently during our experiments.

4.3 The impact of ATR

The TREC 2002 video data set is a very bad collection to test the usability of recognized text for video retrieval. Text detection and recognition algorithms have been developed with different video material in mind such as newscasts, television journals and commercials, where text occurs much more frequently. In newscasts, the frequent appearance of names (interviewed people etc.) and locations rendered in overlay text makes indexing very powerful, especially in combination with face detection.

4.4 The impact of Color

As expected, the color features have been very useful in cases where the query shots where very different from other shots in terms of low level features, or where the relevant shots in the database share common color properties with the example query. For example, if they have been shot in the same environment.

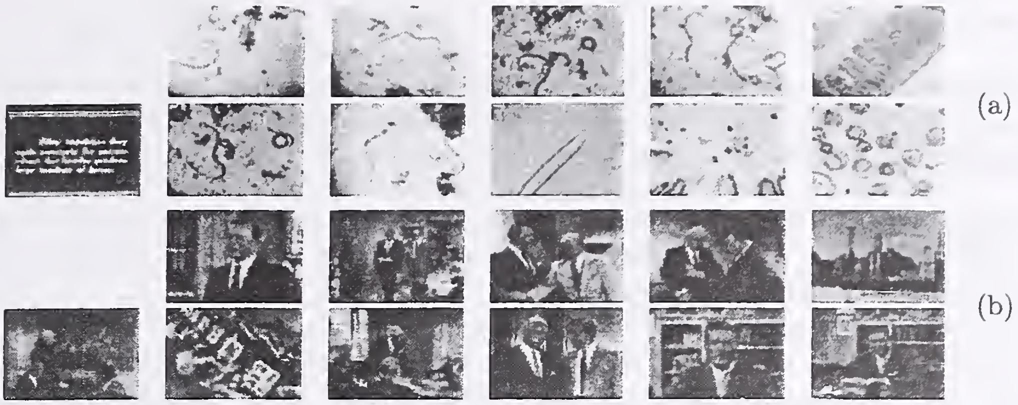


Figure 5: The results of color queries using example shots: searching for living cells under a microscope (a) searching for shots showing James H. Chandler (b).

text/speech	*swimming*	2	OR	10000	AND	2	OR
text/speech	*shore*	1					
Landscape ≥ 0.3 Cityscape ≤ 0.5 Outdoors ≥ 0.5				1			
text/speech	*water*	2	OR			1	
People≤ 0.5 Outdoors ≥ 0.5 Cityscape≤ 0.05 Indoors ≤ 0.75 Landscape≥ 0.5		1					

Table 2: The submitted query formulations for the interactive run of topic 79 *"People spending leisure time on the beach"*

Figure 5a, shows the result set of a color query on an example shot showing living cells under a microscope as requested for TREC search topic 97.

As another example of success, Figure 5b shows the results on the color queries on example shots showing James H. Chandler. Due to the special brownish color of the videos showing Mr. Chandler, the queries were very successful in retrieving shots from the same set of videos. In general, however, the color features need to be used very carefully, since no semantic meaning whatsoever is attached to them.

4.5 Discussion of the experiments

As stated above, the usefulness and impact of the different queries highly depends on the topics and on the example images and videos. In general, the speech recognition transcripts proved to be very important, although queries on them produced very heterogeneous outputs. The color features were only useful in isolated cases, where the targeted shots had very similar low level characteristics to the query images or videos. As expected, the binary features were very useful for filtering queries on the other features but not very useful as only features. Furthermore, the different donated features for the same feature type were not very correlated and tended to have a high amount of noise.

Table 4.5 shows an example of an interactive query submitted. The user realized, that the color features are not very useful in this case and concentrated on queries on the text and speech transcripts using different and more diverse keywords, filtered by binary features whose boundaries have been found also using the graphical browsing tool.

5 Conclusion and future work

A major emphasis in our work was put on the user and the experimental part of the work. We presented a query tool retrieving shots from user requests and a graphical tool which permits to browse through the feature and query space. Our future research will include:

- Exploiting the temporal continuities between the frames, e.g. as already proposed by the Dutch team during TREC 2001[10]. This seems to be even more important using overlay OCR results, since text is often rendered on neighboring shots.
- Combining the binary features (normalization, robust outlier detection etc.).
- Improving the graphical browsing interface. As our experiments showed, quick access to the keyframes and to the movie itself is a keystone in the success of the browsing tool. A browser showing tiny and enlargeable keyframes in the viewing grid instead of video objects should remarkably improve the performance of interactive queries.
- Adding additional features: e.g. explicit color filters, query by sketched example, motion, etc.

The most promising research in video retrieval points into the direction of the use of different features in order to attack the semantic gap from a materialistic point of view. Techniques combining different sources seem to be very successful [7, 3, 8]. However, sophisticated methods combining the large amounts of features are still missing and more theoretical research needs to be done in this area.

References

[1] C.M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, 1996.

[2] R. Duin. The combining classifier: to train or not to train? In IEEE Computer Society, editor, *Proceedings of the ICPR 2002*, volume 2, pages 765–770, August 2002.

[3] G.E. Hinton. Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks*, volume 1, pages 1–6, 1999.

[4] N. Huijsmans and N. Sebe. Extended Performance Graphs for Cluster Retrieval. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 26–32, 2001.

[5] J. Kittler, M. Hatef, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, March 1998.

[6] C.C. LuVogt. Learning to listen to the right information retrieval system. *Pattern Analysis and Applications*, (5):145–153, 2002.

[7] K. Messer, W. Christmas, and J. Kittler. Automatic sports classification. In IEEE Computer Society, editor, *Proceedings of the ICPR 2002*, volume 2, pages 1005–1008, August 2002.

[8] M.R.Naphade and T.S.Huang. Semantic Video Indexing using a probabilistic framework. In *Proceedings of the ICPR 2000*, pages 83–88, 3 September 2000.

[9] M. Rautiainen and D. Doermann. Temporal color correlograms in video retrieval. In IEEE Computer Society, editor, *Proceedings of the ICPR 2002*, volume 1, pages 267–270, August 2002.

[10] The Lowlands Team. Lazy users and automatic video retrieval tools in (the) lowlands. In NIST, editor, *The Tenth Text REtrieval Conference, TREC 2001*, pages 159–168, 2001.

[11] I.H. Witten, A. Moffat, and T.C. Bell. *Managing Gigabytes - Compressing and Indexing Documents and Images.* Academic Press, San Diego, 2nd edition, 1999.

[12] C. Wolf, J.-M. Jolion, and F. Chassaing. Text localization, enhancement and binarization in multimedia documents. In *Proceedings of the ICPR 2002*, volume 2, pages 1037–1040, August 2002.

UMass at TREC 2002: Cross Language and Novelty Tracks

Leah S. Larkey, James Allan, Margaret E. Connell, Alvaro Bolivar, and Courtney Wade

Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA USA 01003

{larkey | allan | connell | alvarob | cwade}@cs.umass.edu

The University of Massachusetts participated in the cross-language and novelty tracks this year. The cross-language submission was characterized by combination of evidence to merge results from two different retrieval engines and a variety of different resources – stemmers, dictionaries, machine translation, and an acronym database. We found that proper names were extremely important in this year's queries. For the novelty track, we applied variants of techniques that have been employed for other problems. In addition, we created additional training data by manually annotating 48 additional topics.

1. Cross Language Track

We submitted one monolingual run and four cross-language runs. For the monolingual run, the technology was essentially the same as the system we used for TREC 2001. For the cross-language run, we integrated some new elements into the Arabic system - a light stemmer that was the result of extensive research [10], the standard probabilistic dictionary based on the UN bilingual lexicon, an expanded dictionary, acronym expansion [12], language modeling, and relevance modeling. In addition, we utilized combination of evidence extensively.

Because our submitted runs were the results of combination of evidence (combining ranked lists from multiple IR runs) we use the term *sub-run* to refer the individual component runs before combination. We first describe the resources, techniques, and models we used, then we describe each run in detail. After presenting our official results, we include some post-hoc runs which correct the errors made in preparing our official submissions, and which help interpret some of the results.

1.1. Arabic and English Resources

The Arabic resources consist of a corpus, normalization and stemming algorithms, and bilingual lexicons. Some of these were developed at UMass and others were provided for TREC participants this year. The English resources consist of a corpus, stemmers, and an acronym facility, as follows:

Arabic Corpus – was the same AFP ARB collection of 383,872 documents in Arabic for TREC-2001 [6] . It was converted to CP1256 encoding, and then indexed in two different ways (1) using UMass normalization and stemming, and (2) using TREC standard normalization and stemming. Only stemmed tokens longer than one byte in length were indexed, and stop words were removed.

UMass Normalization and Light Stemming – These are described in detail in SIGIR02 [10]. In brief, normalization consists of converting to Windows Arabic encoding (CP1256), if necessary, removing punctuation, diacritics, and non-letters, replacing أ, إ, and آ with bare alif ا, replacing final ى with ي, and replacing final ة with ه. Our light stemming algorithm, *light10_stop*, a slight modification of the light stemmer described in [10], consists of stop word removal, stripping definite articles (ال، وال، بال، كال، فال، لل) and و (*and*) from word beginnings and stripping 10 suffixes from word ends (ها، ان، ات، ون، ين، يه، ية، ه، ة، ي). Stopwords were removed using a list of 168 words from Shereen Khoja [8] .

UMass bilingual lexicon – This dictionary is an enhanced version of the dictionary we used for TREC-2001, which was gathered from several sources, as described in [11]. This year we collected more translations from online sources, by including the NMSU proper name dictionary, which used to be available from their Arabic tools page: http://crl.nmsu.edu/~ahmed/downloads.html. In addition, we submitted all the query words from our expanded English queries to two online translation systems, Al Misbar (http://www.almisbar.com/salam_trans.html) and

Ajeeb's Tarjim (http://tarjim.ajeeb.com/ajeeb/). These systems gave one translation (or transliteration) for each term. These were added to the dictionary.

TREC Normalization and Stemming – A Perl script written by Kareem Darwish and modified by Leah Larkey was distributed as the standard stemmer for TREC. It is also a light stemmer, but it is not as light as the UMass stemmer.

TREC bilingual lexicon – We used the bilingual lexicon with probabilities from BBN, derived from the UN parallel corpus. The English words in this dictionary are stemmed with the Porter stemmer, and the Arabic words are stemmed with the TREC standard stemmer. We did not use the bilingual dictionary from Tufts University.

English Corpus – We used AP news articles from 1994 through 1998 in the Linguistic Data Consortium's NA News corpus for an English background language model, and for English query expansion.

English stop words and stemming - English stop words are from INQUERY's standard list of 418 stop words. English stop phrases are defined by regular expressions in a script we have used before in TREC (in English). In order to use the BBN probabilistic dictionary we performed Porter stemming on the queries. In sub-runs that did not use the BBN dictionary, we left the English unstemmed, stemming with *kstem* [9] only when the English word was not found in the dictionary.

Acronym Expansion – In the cross-language runs, we looked up in the *Acrophile* system [12] any sequences of all-capital letters. The top-ranking expansion for each acronym was added to the query.

1.2. Information Retrieval

We used INQUERY [5] for the monolingual run and two of the cross-language sub-runs, and language modeling (LM) for the rest of the crosslingual sub-runs. For both English and Arabic, text was broken up into words at any white space or punctuation characters. For Arabic, there were five additional Arabic punctuation characters included in the definition of punctuation. Words of one-byte length (in CP1256 encoding) were not indexed.

INQUERY - The monolingual run and some of the cross-language runs used a version of INQUERY as the search engine. This version computes the belief function reported in UMass's TREC9 report [2]. The main difference between this version and "real" INQUERY is that proximity information is not stored in the index, so that INQUERY operators requiring proximity information are not implemented.

For cross-language INQUERY retrieval, English queries are translated to Arabic as follows. For each English word in a query, do the following: find the set of all translations in the dictionary. If the English word is not found, stem the English word using the *kstem* stemmer and look it up again. Stem the Arabic translations. If any of the translations consist of an Arabic phrase rather than a single word, enclose the phrase in a `#filreq` operator. Enclose all the alternative Arabic translations for a single English word under a `#syn` (synonym) operator. Finally, take all the `#syn` sets and build a weighted sum query out of all the stemmed translations of the query terms by subsumimg all the synonym sets under a `#wsum` (weighted sum) operator. Each synonym set was given the weight described in the query expansion section.

Crosslingual Language Modeling (LM) - In language modeling, documents are represented as probability distributions over a vocabulary. Documents are ranked by the probability of generating the query by randomly sampling the document model. The language models here are simple unigram models, similar to those of [14]. Unigram probabilities in our official run were estimated as a mixture of maximum likelihood probability estimates from the document and the corpus, as follows:

$$P(Q_e \mid D_a) = \prod_{e\in Q_e}\left((.7)\sum_{a\in Arabic} P(a \mid D_a)P(e \mid a) + (.3)P(e \mid GE)\right)$$

where $P(Q_e|D_a)$ is the probability of generating the query from the document model, the e's are the English words in the query, $P(a|D_a)$ is the probability of an Arabic word a in the document D_a, $P(e|a)$ is the translation probability of seeing the English query word e given the presence of Arabic word a, and $P(e|GE)$ is the probability of the English query word in the English background model. $P(a|D_a)$ is estimated as $tf_{a,Da}/|D_a|$ where $tf_{a,Da}$ is the number of occurrences of term a in Arabic document D_a and $|D_a|$ is the length of document, that is, the number of total term occurrences in the document. The translation probability $P(e|a)$ comes from the bilingual lexicon. If the lexicon was derived from a parallel corpus, these probabilities represent the proportion of the time that Arabic word a was aligned with English word e in the parallel corpus. If the lexicon is a dictionary, and Arabic word a has n different translations into English $e_1, \ldots e_n$, then $P(e|a)$ is estimated as $1/n$. The background probability $P(e|GE)$ is estimated as $P(e \mid GE) = df_{e,C} \Big/ \sum_{t \in C} df_{t,C}$ where $df_{e,C}$ is the number of English documents in C containing term e, and the summation is over all the terms in the English collection, as in [7].

Query Expansion - We expanded English queries in some of our cross-language sub-runs using the AP news articles from 1994 through 1998 in the Linguistic Data Consortium's NA News corpus. This corpus was indexed without stemming, but normalized to lower case. We retrieved the top 10 documents for each query. Terms from these documents received an expansion score which was the sum across the ten documents of the INQUERY belief score for the term in the document. The 5 terms with the highest expansion score were added to the query. Final term weights were set to $2w_o + w_e$ where w_o is the original weight in the unexpanded query and w_e=1.

Arabic query expansion was handled in different ways for INQUERY sub-runs and LM sub-runs. For INQUERY sub-runs, Arabic query expansion was just like English query expansion, except the top 10 documents were retrieved from the Arabic corpus, rather than the English corpus, and 50 terms, not 5, were added to the query.

In language model sub-runs, query expansion was carried out using relevance modeling [13]. The best matching fifty documents were retrieved and 500 words were selected as the new query. Associated with each word is an estimated probability of observing this word in the relevant documents.

Combination of Evidence – Sub-runs were combined into submitted runs by normalizing document scores in ranked lists, and then summing lists two at a time. Score normalization was a linear min-max normalization where $score_{norm}$=$(score\text{-}min)/(max\text{-}min)$.

1.3. Monolingual run

Our one monolingual run was designated UMassM, and carried out as follows:

1. Convert queries to CP1256 encoding
2. Extract titles and descriptions from topics.
3. Remove stop phrases, using a script developed for TREC2001.
4. Stem the query with the UMass light stemmer and remove stop words.
5. Expand the query by adding the best 50 words from the top ranking 10 documents.
6. Retrieve the top 1000 documents using INQUERY.

Overall average precision on this run was .3619. The per-query comparison among the 18 submitted monolingual runs suggests that our approach favored recall over precision. We performed at or above the median average precision on 34 of the 50 queries, and below the median on 16 queries. In number of relevant documents retrieved in top 1000, we were at or above the median in 44 or the 50 queries, and at the highest number in 28 queries.

1.4. Cross-language runs

Our cross-language submissions relied heavily on combination of evidence this year. We used two different dictionaries that could not be combined because they were built with different stemmers. In our own dictionary, a composite of a variety of different sources, English was not stemmed, and Arabic words were stemmed using the UMass light stemmer. In the standard probabilistic dictionary, (the bilingual lexicon built at BBN using the UN parallel corpus), Arabic words were stemmed using the Darwish stemmer, and English was stemmed with the Porter stemmer. We ran several independent retrieval sub-runs and combined their ranked lists at the end.

Each sub-run used one of two different retrieval engines – INQUERY or LM, one of two different resource sets: UMass dictionary and stemmer, or TREC standard probabilistic dictionary and stemmer, one of 3 different query expansion options: English only, Arabic only, or English and Arabic, and one of two different selections from the topic: title and description, or title, description, and narrative.

The steps were as follows: (For sub-runs)

1. Select text from titles and descriptions, or titles, descriptions, and (in X2n and X6n conditions) select capitalized words from narrative field.
2. Remove stop phrases from English queries.
3. Expand acronyms
4. Lower case the query

5. Expand the English query (optional – depends on condition)
6. Expand the Arabic query (optional – depends on condition)
7. Retrieve top 2000 documents

Twelve different crosslingual conditions were run. These sub-runs were combined into four cross language runs that can be briefly described as follows:

1. **UMassX2** – combination of 2 INQUERY cross language sub-runs both using title and description fields.
2. **UMassX6** – combination of all 6 cross language sub-runs using title and description fields.
3. **UMassX2n** – combination of 2 INQUERY sub-runs using title, description, and narrative fields.
4. **UMassX6n** – combination of all 6 cross language sub-runs using title, description, and narrative fields.

Table 1 lists the resources used in each of the 12 sub-runs, and indicates which sub-runs composed each submitted run. Names of submitted runs are abbreviated e.g. *UMassX2* as *X2*, etc.

Table 1: Definition of sub-runs

Sub-run	Component of	Engine	Dictionary + Stemmer	Expansion	Narrative Included?
Inq-UM-EngAr	X2, X6	INQUERY	UMass	English+Arabic	no
Inq-TREC-EngAr	X2, X6	INQUERY	TREC	English+Arabic	no
LM-UM-Ar	X6	LM	UMass	Arabic	no
LM-UM-Eng	X6	LM	UMass	English	no
LM-TREC-Ar	X6	LM	TREC	Arabic	no
LM-TREC-Eng	X6	LM	TREC	English	no
Inq-UM-EngAr-Nar	X2n, X6n	INQUERY	UMass	English+Arabic	yes
Inq-TREC-EngAr-Nar	X2n, X6n	INQUERY	TREC	English+Arabic	yes
LM-UM-Ar-Nar	X6n	LM	UMass	Arabic	yes
LM-UM-Eng-Nar	X6n	LM	UMass	English	yes
LM-TREC-Ar-Nar	X6n	LM	TREC	Arabic	yes
LM-TREC-Eng-Nar	X6n	LM	TREC	English	yes

1.5. Cross-Language Results

Table 2 summarizes the results of our official submissions. Combination of results based on different resources improved performance. Inclusion of narrative words also improved performance a great deal. UMassX6n had the highest mean average precision of all the officially submitted cross language runs in TREC 2002.

Table 2: Monolingual and Cross-language Results: official runs

Name of Run	Official Mean Average Precision	Number of Queries at or Above Median Average Precision	Number of Queries at or Above Median Rel Ret in top 1000	Post hoc Average Precision
UMassM	.3619	34/50	44/50	.3619
UMassX2	.3538	30/50	40/50	.3589
UMassX6	.3658	36/50	42/50	.3801
UMassX2n	.3900	35/50	37/50	.3941
UMassX6n	.3996	39/50	41/50	.4107

1.6. Post hoc Cross-Language Experiments

Post hoc experiments were performed first, to correct some errors in our official submissions, second, to provide a "standard resources" run, and third, to explore the role of acronyms, proper names, and stemming of the UN parallel corpus.

1.6.1. Fixing Errors

We discovered two problems after submitting our results. First, the Porter stemmer used in processing the BBN bilingual corpus was different from our version of the Porter stemmer, so that many English words were not found in the probabilistic dictionary. For example, *contrary* was stemmed to *contrari* by the BBN Porter stemmer, but to *contrar* by our Porter stemmer. *Money* became *monei* under the BBN Porter stemmer, but remained *money* under our Porter stemmer. When we reran the sub-runs with compatible Porter stemming, results on those sub-runs improved, as did the combinations that included them.

A second problem resulted from a procedural error, in which the language model runs (but not the INQUERY runs) using the UMass resources were run with an older, smaller version of the dictionary. We therefore reran the affected conditions with the correct dictionary, and obtained the results shown in the last column of Table 2.

Overall, the greater coverage of the dictionaries and the use of compatible versions of the Porter stemmer improved performance. The overall patterns remained the same - combination of resources improved performance, and retrieval was more effective when the narrative portion was included in the query.

1.6.2. Standard Resources Run

Although we did not submit an official standard resources run we ran one later for this report. Recall that one of the two sub-runs that made up **UMassX2 and UMassX2n**, and three of the six sub-runs that made up **UMassX6** and **UMassX6n**, used the standard parallel corpus dictionary and stemmer. The ***Standard Resources*** column of Table 3 shows the results when the sub-runs based on the UMass resources and acronym expansion were excluded. Only the sub-runs based on the standard resources were included. Thus, in the ***Standard Resources*** column, the **UMassX2** and **UMassX2n** rows show the results of a single sub-run, and **UMassX6** and **UMassX6n** rows each show results based on a combination of three, rather than six, sub-runs. Relative to these three way combinations, the additional resources increased average precision 3 percentage points for title+description queries, and 4 points for title+description+narrative queries.

1.6.3. Acronym expansion

Because this was the first time acronym expansion was used in the TREC cross-lingual track, we assessed its contribution separately. We reran the **UMassX6** and **UMassX6n** runs without expanding acronyms. The results, shown in the ***No Acro*** column of Table 3, revealed that acronym expansion added almost nothing. Analysis of individual queries containing acronyms revealed that while acronym expansion helped on some queries, it hurt on others.

1.6.4. Importance of Proper Names

It had struck us in informal query analyses that proper names were extremely important in these queries. We hypothesized that successful retrieval depended upon having these proper names in the lexicon. In order to test this, we identified all the proper names (people, places, organizations, acronyms) in our queries and in our expanded English queries, and made special versions of the UMass and BBN dictionaries from which these names had been removed. The column labeled *No Names* in Table 3 gives the results. Performance dropped more than 50% when names were not available in the dictionaries. We speculate that one reason dictionaries derived from parallel corpora work so well is that they cover so many more proper names than do static dictionaries.

Table 3: Post hoc average precision: Cross Language runs

Name of Run	Official	Errors Corrected	Standard Resources	No Acro	No Names	Reprocessed UN Corpus
UMassX2	.3538	.3589	.3141	.3577	.1561	.3668
UMassX6	.3658	.3801	.3508	.3795	.1629	.3948
UMassX2n	.3900	.3941	.3337	.3936		.3983
UMassX6n	.3996	.4107	.3724	.4104		.4189

1.6.5. Reprocessing the UN Corpus

Although the standard parallel corpus dictionary as processed by BBN was a very valuable resource, we found it awkward to fit into the rest of our system because of the Porter stemming used on the English words, and because of the small differences between the standard Arabic stemmer and the UMass light stemmer. We reprocessed the UN corpus, obtained from LDC, using the same configuration Alex Fraser used at BBN, except we used different preprocessing in preparing the English and Arabic input to GIZA++. English words were lower cased, and stop words were removed. Arabic words were stemmed and stop words removed using the UMass (*light10_stop*) stemmer. Retrieval results comparing the two versions of the parallel corpus dictionary can be seen in Table 4. Table 4 contains only sub-runs and combinations that used the parallel corpus dictionary. It does include any runs that used the UMass dictionary. The improvement in the official runs that include all the resources can be seen in Table 3, above, in the ***Reprocessed UN Corpus*** column.

Table 4: Mean average precision using two dictionaries made from the UN parallel corpus.

Type of Run		TREC Standard Dictionary	Reprocessed UN Corpus
INQUERY	title+description	.3161	.3413
LM	title+description	.3464	.3637
INQ + LM	title+description	.3520	.3678
INQUERY	title+desc+narrative	.3350	.3614
LM	title+desc+narrative	.3605	.3804
INQ+LM	title+desc+narrative	.3734	.3880

2. Novelty Track

Our attempts to find relevant and novel information focused on variants of techniques that have been employed for other problems [1] [3] [4]. Basically, we looked for relevant sentences by comparing them to the query, and we looked for redundancy by estimating whether a sentence was dissimilar from all prior (relevant) sentences. We found that the task of recognizing relevant sentences was the major challenge and that our errors there account for poor overall performance.

One notable feature of the CIIR's participation in the novelty track is our creation of additional training data. We hired students to annotate 48 topics in addition to the handful provided by NIST. Details on that process are discussed below.

2.1. Creating additional training topics

For the purpose of this evaluation we built our own collection of documents and sentence level relevance judgments. We randomly chose 48 topics from the TREC-7 and TREC-8 ad-hoc retrieval tracks (topics 300-450)—though we were careful not to use the topics set aside for the novelty track evaluation. For each of the 48 topics, we used the INQUERY search engine to find the top-ranked documents in a subset of TREC volumes 4 and 5 (Federal register 1994, LA Times 1989-90, and FBIS 1996). We selected the top 25 known-relevant documents (based on TREC judgments) for each topic.

We followed the same methodology defined by the novelty track to collect relevance and novelty assessments. We hired undergraduate students, who were otherwise unaffiliated with our research, to read the relevant documents for each topic in rank order. They extracted relevant and novel sentences from each topic in a two step process. First, the assessors were told to read a printed copy of the relevant documents and highlight the relevant sentences. Second, they read the sentences marked as relevant again and flagged the ones that contained novel information. For this process, order is very important. By definition, a sentence is novel if it provides totally new information or further details on previously seen information. Instances that summarize details seen earlier in the document stream were not considered novel.

Some statistics for the constructed training data are presented in Figure 1 and Table 5. These statistics are consistent with the NIST-provided training topics as well as the evaluation topics. Interestingly, for the average topic less than 5% of the sentences contain relevant material. Further, more than 80% of the relevant sentences on average contain novel information. That is, most of the material is non-relevant and most of the relevant material is novel. The material for reconstructing our data and some additional statistics about the data are available at [http://ciir.cs.umass.edu/downloads/access.html]. That material assumes that TREC volumes 4 and 5 are available to the user.

Table 5: Collection statistics summary for training set

TOPICS	# ASSESSORS	DOCS	SENT	NOVEL	REL	REL/SENT	NOVEL/REL
48	6	1122	84588	2759	3400		
	AVERAGE/TOPIC	23.38	1762	57.48	70.83	4.6%	81.6%
	σ	3.923	797.22	46.217	55.63	3.64%	12.74%

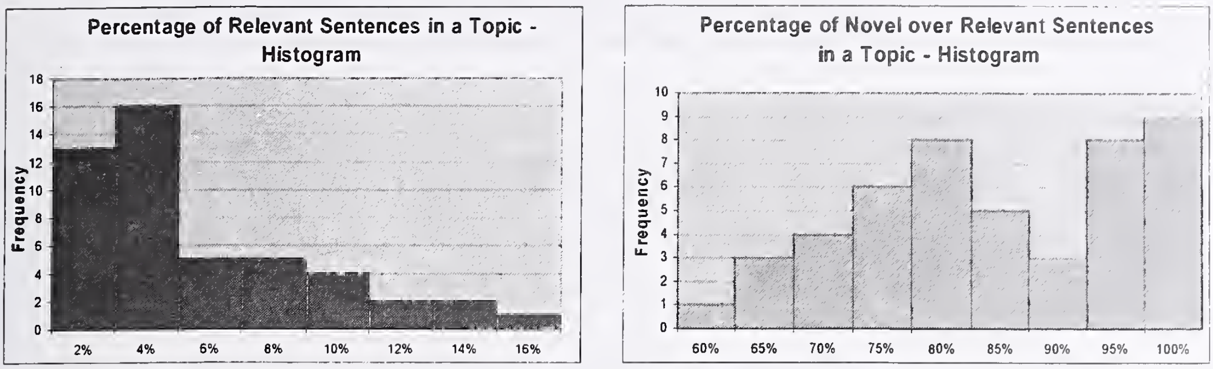


Figure 1: Histograms of distribution of (a) the percentage of relevant sentences over the total number of sentences and, (b) the percentage of novel sentences in terms of the number of relevant sentences.

We were curious about the impact of time on a person's conception of novelty. For instance, we were interested in answering question like the following: How far could an assessor go without losing track of the mental representation of a novel sentence with respect to a particular topic? What is the behavior of the novelty rate as more documents are added to the knowledge base? To answer this questions, one of the assessors was told to read through a bigger set of documents (75 documents) carrying out the relevance and novelty judgments steps as they were explained before. The topic chosen for this experiment was 422 (Figure 2). The results are presented in Table 6.

```
<top>
  <num> Number: 422
  <title> art, stolen, forged
  <desc> Description:
  What incidents have there been of stolen or forged art?
  <narr> Narrative:
  Instances of stolen or forged art in any media are relevant.  Stolen mass-  produced
things, even though they might be decorative, are not relevant (unless they are mass-
produced art reproductions). Pirated software, music, movies, etc. are not relevant.
</top>
```

Figure 2: Topic 422's description.

Table 6: Annotation information for topic 422. The first row presents statistics for an assessor who was asked to annotate 75 documents rather than just 25.

TOPICID	ANNOT	DOCS	SENT	NEW	REL	REL/SENTS	NEW/REL
422	D	75	3593	164	181	5.0%	90.6%
422	D	25	2065	89	94	4.6%	94.7%
422	C	25	2065	75	77	3.7%	97.4%

The assessor in charge of this task reported that it was not difficult to assess the relevance or novelty of a sentence even with this many documents, because it was fairly easy to maintain a clear sense of the topic definition and when new information about the topic was appearing.

Table 6 shows an increase for the percent of relevant sentences across different assessors and different evaluation set sizes. The difference is presumably accounted for by normal inter-annotator disagreement. Interestingly, though, the proportion of new material decreases with more sentences judged. Our intuition tells us that as more documents are processed and our knowledge base about the topic increases, the tendency to find new sentences should be greatly reduced: the topic should be fully covered. The results from trying one topic in more detail might be explained by a lack of redundancy in the collection or by the intrinsic property of the topic to constantly generate new events related to the same topic (and thus novel sentences). Topics defined the way topic 422 is defined admit constant generation of new events that must create new relevant and novel sentences.

We were also concerned by the low number of relevant sentences and wondered if this was the result of the annotation instructions. We told one of our assessors to read the relevant documents for topic 327 and topic 417 (Figure 3) and identify the sentences that do *not* provide any relevant information whatsoever in relation to the topic. The results are presented in Tables 7 and 8.

```
<top>
<num> Number: 327
<title> Modern Slavery
<desc> Description:
Identify a country or a city where there
is evidence of human slavery being
practiced in the eighties or nineties.
<narr> Narrative:
A relevant document would present
evidence of current slavery practices
being carried out.  It would identify a
specific country or city and give some
information on who the slaves are, or who
was buying or selling them, and for what
purposes they were being used.
References to slavery being carried out
several years ago would not be relevant.
</top>
```

```
<top>
<num> Number: 417
<title> creativity
<desc> Description:
Find ways of measuring creativity.
<narr> Narrative:
Relevant items include definitions of
creativity, descriptions of
characteristics associated with
creativity, and factors linked to
creativity.
</top>
```

Figure 3: Topic description for topic 327 and topic 417

Table 7: Comparison of assessor results when one assessor located relevant sentences and the other locates non-relevant sentences. Upper right and lower left portions of each table represent disagreement

	TOPIC 327	ASSESSOR D			
		NOT JUDGED		NOT REL	
ASSES. F	RELEVANT	19	4.2%	40	8.8%
	NOT JUDGED	8	1.8%	389	85.3%

	TOPIC 417	ASSESSOR D			
		NOT JUDGED		NOT REL	
ASSES. A	RELEVANT	20	0.9%	9	0.4%
	NOT JUDGED	103	4.4%	2215	94.4%

Table 8: Comparison of sentences judged by two assessors in selected topics

TOPICID	ASSESSOR	*		*/S
327	A	59	Relevant	12.9%
	D	429	Not Relevant	94.1%

Documents: 11 Sentences: 456

TOPICID	ASSESSOR	*		*/S
417	F	29	Relevant	1.2%
	D	2224	Not Relevant	94.8%

Documents: 25 Sentences: 2347

Table 7 shows the inconsistencies between two assessors, one finding relevant and one finding non-relevant. For topic 327, for example, the upper-right cells indicate that 40 sentences were explicitly identified as relevant *and* as non-relevant. Presumably those reflect inter-annotator disagreement, though we have not adjudicated the results to see whether there are errors. Nineteen of the sentences were judged relevant by both approaches, and almost 86% of the sentences were consistently judged non-relevant.

Table 8 shows for topic 327 that an assessor finding relevant sentences found that 12.9% of the sentences were relevant, whereas an assessor looking for non-relevant implicitly found only 5% of the sentences relevant. The difference is surprising, particularly since we were expecting that the "find non-relevance" assessor would "find" substantially more relevant material. On the other hand, for topic 417 we obtained a more reasonable result with a very low 0.4% of inter-annotator disagreement. These differences among topics may be explained by the information need that the topic describes, which for topic 327 seems to be pretty ambiguous.

Similar experiments on a larger collection of topics might make clearer what is happening.

2.2. Experiments finding relevant sentences

In order to extract the sentences with relevant information, we used the traditional IR ranking approach with three different retrieval models. Pseudo-relevance feedback (PRF) was used with each approach and executed depending on the retrieval model being used.

1. *TFIDF*. Here, we tried the traditional TFIDF approach. Given a query q and sentence s:

$$score(s) = \sum_{t \in q} \log(tf_{t,q}+1)\log(tf_{t,s}+1)\log\left(\frac{n+1}{.5+sf_t}\right)$$

   $tf_{t,q}$ and $tf_{t,s}$ are the number of times term t occurs in the query and sentence, respectively, sf_t is the number of sentences in which term t appears, and n is the number of sentences in the collection.
2. ***Simple Language Modeling (KLD).*** Given smoothed language models of a query q and a sentence s, the score is given by the Kullback-Leibler divergence (KLD) between the two mass distributions:

$$score(s) = KLD(q \| s) = \sum_{w \in V} p(w \mid q) \cdot \log \frac{p(w \mid q)}{p(w \mid s)}$$

3. ***Two-Stage Smoothing in Language Modeling.*** The two-stage smoothing method allows the use of different smoothing techniques for the query and the sentence language models. This is used in order to differentiate the two roles that smoothing plays in the retrieval process. For the sentence, smoothing assigns non-zero probabilities to words not present in the sentence. For the query, smoothing "explains away" the common non-topic words in the query. This approach is extensively explained in [15].

Multiple runs were carried out on our training topics in order to tune the multiple parameters in the different retrieval models. Results from the best runs are presented in Figure 4. Although TFIDF with PRF had the best average precision, its performance is not significantly different (student's t-test) from the other models. For all models, performance of the retrieval process at sentence level was poor and very hard to improve.

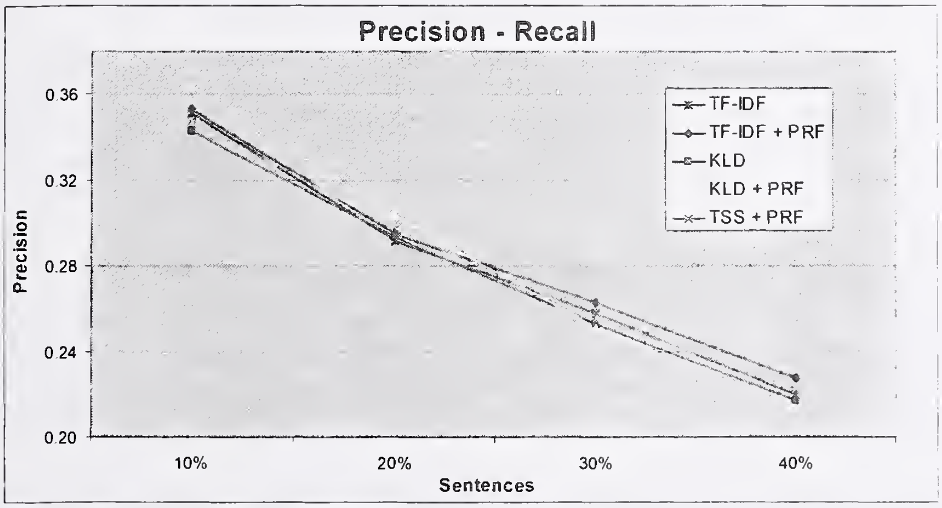


Figure 4: Results for sentence relevance retrieval on the training topics

We also explored the influence of query length on performance. We used "short" queries and "long" queries. For every case, we tried using only some portion of the topic description. For short queries we tried using only the topic title or the topic title and description. As long queries, we tried using the topic title, description, and narrative. Our results were consistent with the results reported in [15]. On average, it is better to use the complete topic text as opposed to using only portions of it.

Some additional experiments were carried out, without much success, to try to improve performance by means of query preprocessing. A standard ad-hoc query algorithm [2] was run on the query text (topic text) for the different topics. The goal of this algorithm was to get rid of what we believed were query stop words and stop phrases. We believed that words like *"narrative"* and *"description"* as well as phrase patterns such as *"A document that discusses word [word ... word] is considered non-relevant"* should not be present in the query. Contrary to our intuition, the use of this algorithm did not improve the retrieval performance significantly.

2.3. Looking for novelty

Our novelty detection systems take as input the ranked list of relevant sentences for each topic (i.e., the output of the previous step). We decided to use the vector space retrieval model with TFIDF weighting and pseudo-relevance feedback as we found that it was the method that worked best with our training data. Both of our systems (CIIR02tfkl and CIIR02tfnew) use this method to identify the relevant sentences. We determined through trial and error on the training data that our results were best if we used the top 10 percent of relevant sentences, even though we know from the training topics that only about 5% of the sentences are likely to be relevant. These top relevant sentences are re-indexed using Lemur.[1] No stopping or stemming is used at this stage.

Our two systems assign novelty scores to sentences in different ways:

CIIR02tfkl. The CIIR02tfkl system uses the KL-divergence between two language models as its scoring method. For each topic, the documents are considered in the order given by the task and novelty scores are assigned using the following method. The first relevant sentence of the first relevant document is assigned a maximum score. For the remaining sentences we calculate a collection language model and a sentence language model in Lemur:

collLM(i) (the collection language model for sentence i) is a maximum likelihood model built on sentences 1 to (i-1), smoothed using linear interpolation against a maximum likelihood model built on sentences 1 to i.

sentLM(i) (the sentence language model for sentence i) is a maximum likelihood model built on sentence i, smoothed using linear interpolation against a maximum likelihood model built on sentences 1 to i.

Sentence i's score is the KL-divergence between its collection and sentence language models, KLD(sentLM(i) || collLM(i)).

We set the smoothing parameters for both language models so that almost no smoothing occurs. These parameter settings were chosen because they achieved the best results on our training data.

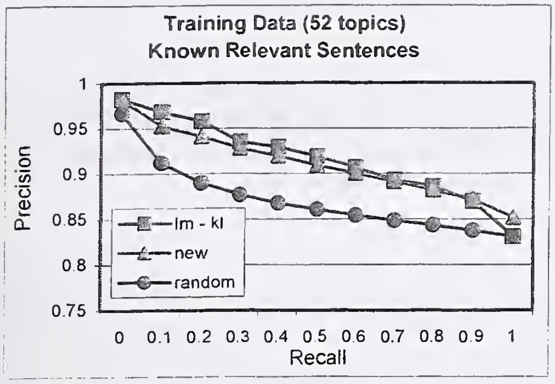
CIIR02tfnew. The CIIR02tfnew system assigns novelty scores in a very simple way. For each topic, it considers the documents in the order specified by the task. Each sentence is treated as a set of words and a sentence's score is equal to the number of new words in the set (i.e., words that have not appeared so far in the sentences for that topic).

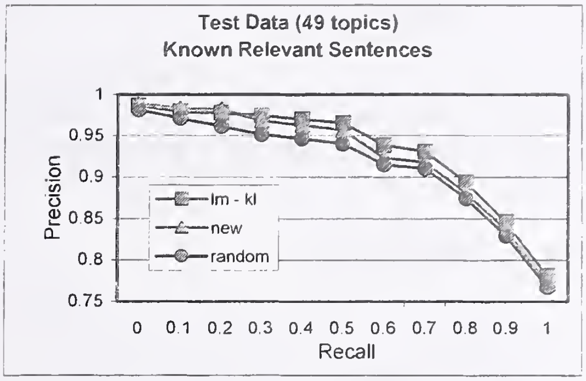
We observed that in our training data, approximately 80 percent of the sentences judged relevant by the annotators were also judged new. Therefore, both of our systems return the top 80 percent of the ranked list of novelty scores as new.

Interestingly, for both the training and test data, when we ran our two systems on the collection of *known* relevant sentences (i.e., we cheated), CIIR02tfkl performed better than CIIR02tfnew. However, when we ran these systems on our own relevance results (i.e., relevance results with errors), CIIR02tfnew performed better than CIIR02tfkl. We hypothesize that non-relevant sentences are likely to be identified as novel and that CIIR02tfkl models novelty better than CIIR02tfnew and therefore tends to pull those non-relevant sentences towards the top of the novelty rankings. We have not sufficiently investigated this issue, however. It may, for example, be nothing more than a statistical anomaly.

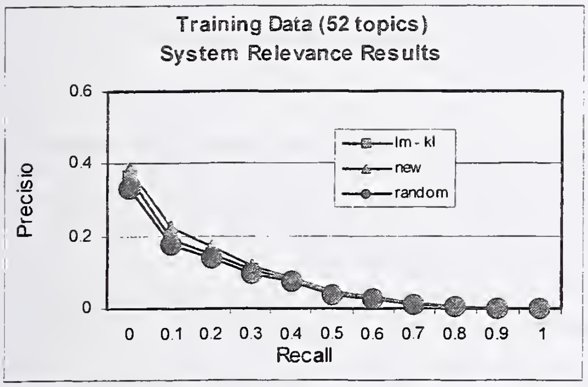
The following graphs show the effectiveness of the two techniques when only relevant sentences are ranked. Note that even a random ranking of these sentences does quite well, because about 80% of them are novel.

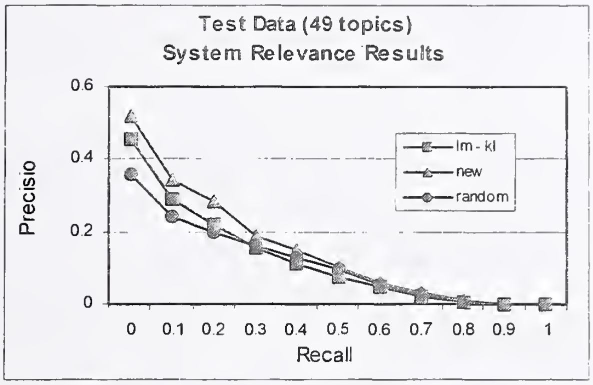
[1] Lemur is a toolkit from CMU and UMass Amherst intended to support the use of language modeling techniques for information retrieval. It is freely available for research purposes at http://www.cs.cmu.edu/~lemur.



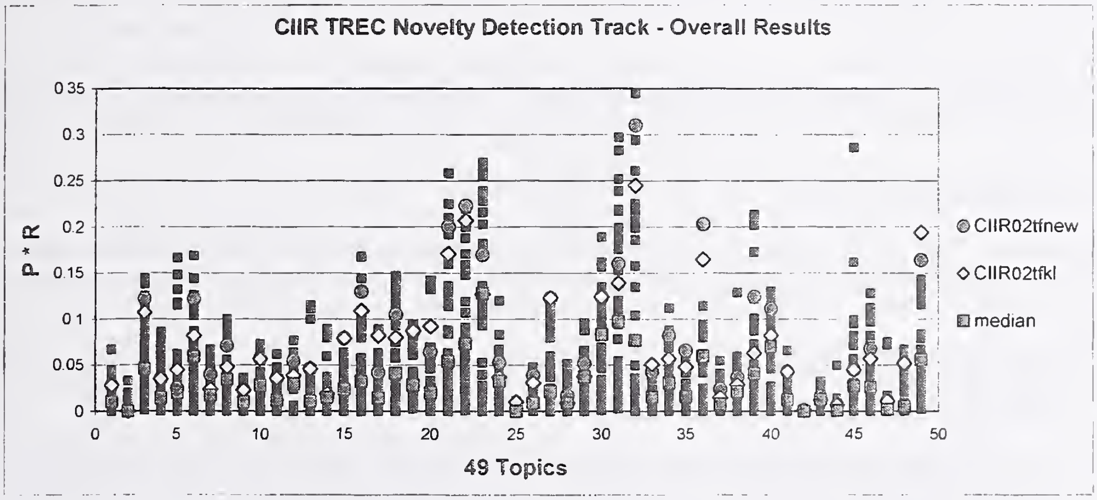


The next graphs provide the same information for when the "relevant" sentences are chosen using one of the approaches described above—i.e., generated by a system. Overall performance drops substantially because the quality of the initial retrieval is poor (the scale of the axes is dramatically different).





This final graph shows how the results of our official submissions compared to other submissions of the TREC and shows on a query-by-query basis how our two approaches stacked up.[2]



[2] This graph compares systems based on precision * recall, the evaluation measure used at the TREC workshop. After the workshop, the evaluation metric was changed to the F measure.

3. Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by SPAWARSYSCEN-SD grant numbers N66001-99-1-8912 and N66001-02-1-8903, and in part by Advanced Research and Development Activity under contract number MDA904-01-C-0984. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

4. References

[1] Allan, J. Introduction to topic detection and tracking. In *Topic detection and tracking: Event-based information organization*, J. Allan, Ed.: Kluwer Academic Publishers, pp. 1-16, 2002.

[2] Allan, J., Connell, M. E., Croft, W. B., Feng, F.-f., Fisher, D., and Li, X. INQUERY and TREC-9. Presented at The Ninth Text REtrieval Conference (TREC-9), Gaithersburg, Maryland, 2000.

[3] Allan, J.. Gupta, R., and Khandelwal, V. Temporal summaries of news topics. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*, pp. 10-18, 2001.

[4] Allan, J., Jin, H., Rajman, M., Wayne, C., Gildea, D., Lavrenko, V., Hoberman, R., and Caputo, D. *Topic-based novelty detection: 1999 summer workshop at clsp.*, Final Report available on-line at http://www.clsp.jhu.edu., 1999.

[5] Callan, J. P., Croft, W. B., and Broglio, J. TREC and TIPSTER experiments with INQUERY. *Information Processing and Management*, 31 (3), pp. 327-343, 1995.

[6] Gey, F. C. and Oard, D. W. The TREC-2001 cross-language information retrieval track: Searching Arabic using English, French, or Arabic queries. In *TREC 2001*. Gaithersburg: NIST, 2002.

[7] Hiemstra, D. and de Vries, A. *Relating the new language models of information retrieval to the traditional retrieval models*. University of Twente, Enschede. The Netherlands, CTIT Technical Report TR-CTIT-00-09. May 2000 2000.

[8] Khoja, S. and Garside, R. *Stemming Arabic text*. Computing Department, Lancaster University, Lancaster, U.K. http://www.comp.lancs.ac.uk/computing/users/khoja/stemmer.ps, 1999.

[9] Krovetz, R. Viewing morphology as an inference process. In *Proceedings of the sixteenth annual inte-rnational ACM SIGIR conference on research and development in information retrieval*, pp. 191-203, 1993.

[10] Larkey, L. S., Ballesteros, L., and Connell, M. E. Improving stemming for Arabic information retrieval: Light stemming and co-occurrence analysis. In *SIGIR 2002: The twenty-fifth annual international ACM SIGIR conference on research and development in information retrieval*. Tampere, Finland: ACM, pp. 275-282, 2002.

[11] Larkey, L. S. and Connell, M. E. Arabic information retrieval at UMass in TREC-10. In *TREC 2001*. Gaithersburg: NIST, 2001.

[12] Larkey, L. S., Ogilvie, P., Price, M. A., and Tamilio, B. Acrophile: An automated acronym extractor and server. In *Digital libraries '00 - the fifth ACM conference on digital libraries*. San Antonio, TX: ACM Press, pp. 205-214, 2000.

[13] Lavrenko, V., Choquette, M., and Croft, W. B. Cross-lingual relevance models. In *SIGIR 2002: The twenty-fifth annual international ACM SIGIR conference on research and development in information retrieval*. Tampere, Finland: ACM, pp. 175-182, 2002.

[14] Xu, J., Weischedel, R.. and Nguyen, C. Evaluating a probabilistic model for cross-lingual information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*. New Orleans: ACM Press, pp. 105-110, 2001.

[15] Zhai, C. and Lafferty, J. Two-stage language models for information retrieval. In *SIGIR 2002: The twenty-fifth annual international ACM SIGIR conference on research and development in information retrieval*. Tampere, Finland: ACM, pp. 49-56, 2002.

Homepage Finding and Topic Distillation using a Common Retrieval Strategy

Vo Ngoc Anh *Alistair Moffat*

Department of Computer Science and Software Engineering
The University of Melbourne
Victoria 3010, Australia

{*vo,alistair*}*@cs.mu.oz.au*

Abstract: For the *TREC-2002* web track the University of Melbourne experimented with a system designed primarily for topic relevance tasks, and applied it directly to the homepage finding and topic distillation tasks. Our intention was to process queries regardless of their classification, as discriminating information may be unavailable in practice. An integer-valued weighting scheme reported in earlier work was employed, modified to take into account anchor text and many of the metadata fields, but not the URL text, and not the link structure information. Our experiments were carried out using a distributed retrieval system, with data spread across a sixteen node cluster. Indexing and query processing is fast, and the total index size is small.

1 Background

Our goal in participating in the Web track this year is to explore ways in which normal techniques for topic relevance tasks can be applied to the specific tasks of homepage finding and topic distillation. We examined variations of topic relevance systems, hoping to determine the relative performance of this approach in comparison with the mechanisms employed by other participants.

Our approach can be briefly characterized as: (a) using realistic queries; (b) using document structure, metadata and anchor text, but not URL text; (c) not using link structure; (d) using simple stemming with case-folding; (e) using a (modified) vector space model with integer-valued quantized impacts replacing floating-point term weights in the similarity computation; and (f) applying a common retrieval strategy to the homepage finding and topic distillation tasks. Techniques like phrase indexing, locating noun phrases, and query expansion were not considered in our experiments.

A number of considerations led to our selection of runs. First, we believe that our impact transformation technique [Anh and Moffat. 2002] can give good retrieval effectiveness, and we wished to validate that optimism in the *TREC* context. The Web tasks this year do not include direct topic relevance, but we have maintained the main characteristics of the impact transformation technique and have applied it directly to the new tasks. We sought to use a common system for both of the tasks, and also for the traditional topic relevance task. In order to defend this position, the only training we performed was using the 2000 topic relevance task, and we did not then tune against the 2001 homepage task results.

Second, although metadata and anchor text are integrated into the weighting scheme, their inclusion is not at the cost of the "common" system, and they are regarded as being part of the document in question. A good retrieval system should make effective use of all sources of information, but should also be able to retrieve matching documents regardless of whether or not they contain metadata and/or structuring elements. On the other hand, our exclusion of URL details from the indexing process was a programming oversight, and we would expect to include whatever information they contain in a future system.

Third, the link structure is not used in our weighting scheme simply because we still have not found an efficient (and effective) way to do it. It is reasonable to expect that using link structure enhances retrieval effectiveness, but has a non-trivial cost in terms of implementation effort.

Finally, although we have no great expectation that our approach leads to superior effectiveness results, we also believe that producing a universal method for Web searching, regardless of any specific task, is a desirable goal. Even if users are able to tag their queries, it seems improbable that many would.

Level	Description
1	The following fields (and no others) are indexed: text content, outgoing anchors, incoming anchors, titles, headings, keywords, descriptions.
2	Same as level *1*, except with outgoing anchors excluded.
3	Only text content is indexed.
4	Same as level *1*, except with text content excluded.
5	Only incoming anchors are indexed.

Table 1: Information content levels of indexes used in the experiments along with their description.

2 Weighting scheme

The weighting scheme employed in our experiments is based on our impact transformation technique [Anh and Moffat, 2002], with some modifications made to alter the within-document frequencies prior to using them in the similarity score computation.

Weighting document components An important aspect of our experiments is the effect that indexing different document fields has on the performance of the system, in terms of both effectiveness and efficiency. The document fields considered include the text content: any outgoing anchor text; any incoming anchor text; the title: any headings: keywords; and description fields. All other fields are ignored when indexing. The combinations of features used in our experiments are listed in Table 1.

Whenever an indexed term t is parsed in a document d, a certain contribution is added to the within-document frequency $f_{d,t}$. The contribution differs according to the field this occurrence of t is in. As a baseline, the contribution to $f_{d,t}$ is always 1 for term occurrences in the document text. A contribution of 8 is used for outgoing anchor text; 8 for terms in incoming anchor text from a different host; 4 for incoming anchor text from the same host; and 2 for any of the other fields listed in Table 1. Note that the setting of contribution weights is a very coarse way of boosting the importance of metadata fields and anchor text, and to date we have not made any attempt to tune the parameters for better effectiveness.

The sum of the contributions of term t in document d is used as a surrogate for the conventional $f_{d,t}$ value in the ensuing similarity computation.

Vocabulary Every word that appears in one or more of the selected fields is case-folded, slightly stemmed by the removal of "s" and "ed" suffixes, and indexed. The intention of the stemming process is to only remove differences between plural and single nouns, and between variant verb appearances for regular cases.

Similarity score The modified similarity score $S_{d,q}$ between a document d and a query q is represented as:

$$S_{d,q} = \sum\nolimits_{t \in q \cap d} \omega_{d,t} \cdot \omega_{q,t}$$

where $\omega_{x,t}$ is an integer in the range 1 to 2^b, with (in these experiments) $b = 5$. The value $\omega_{x,t}$ represents the quantized impact of term t in document or query x, and is calculated in two steps.

First, a normal cosine similarity is used to compute $w^*_{d,t}$ and $w^*_{q,t}$:

$$\begin{aligned}
w_{d,t} &= (1 + \log_e f_{d,t}) \\
W_d &= \sqrt{\sum_{t \in d} w^2_{d,t}} \\
W^*_d &= 1/((1-s) + s \cdot W_d/W^a) \\
w^*_{d.t} &= w_{d.t}/W^*_d \\
w^*_{q,t} &= \log_e\left(1 + \frac{f^m}{f_t}\right) \cdot (1 + \log_e f_{q,t})
\end{aligned}$$

where $f_{d,t}$ and $f_{q,t}$ are the term frequency in the document and query calculated as shown above; f_t is the number of documents that contain term t; f^m is the greatest value of f_t in the collection; W_d is document length; W^a is the average value of W_d over the documents in the collection: and W^*_d represents the normalized document length using pivoted normalization [Singhal et al., 1996] with a slope of $s = 0.7$.

Then, a small enough positive value L and a large enough positive value U are chosen such that all of the $w^*_{d.t}$ lie between L and U, thereby allowing the following transformation to be calculated:

Run	Weighting	Effectiveness		
label	scheme	Reciprocal rank	% in top 10	% unfound
MU106	*1L*	0.576	78.7	7.3
MU609	*3G*	0.524	73.3	16.7
MU80A	*5G*	0.402	53.3	26.7
MU307	*4L*	0.207	31.3	58.0

Table 2: Description of homepage finding runs and their official results.

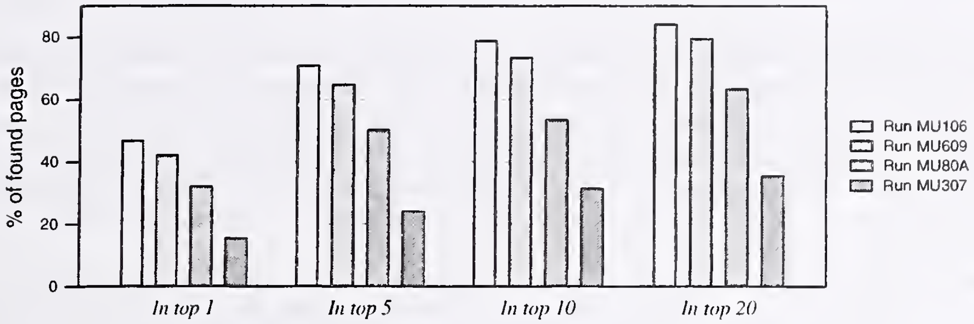


Figure 1: Behavior of weighting schemes for the homepage finding task, percentages of correct homepages found at different number of documents retrieved.

$$\begin{aligned}\omega_{d,t} &= \left\lfloor 2^b \cdot \frac{\log w^*_{d,t} - \log U}{\log U - \log L + \epsilon} \right\rfloor + 1 \\ \omega_{q,t} &= \left\lfloor 2^b \cdot \frac{\log w^*_{q,t} - \log U}{\log U - \log L + \epsilon} \right\rfloor + 1\end{aligned}$$

in which $B = (U/L)^{L/(U-L)}$, and ϵ is a small positive quantity, and the impact values are recorded and used as integers.

Our experiments made use of two different types of transformation, characterized by the choice of L and U. In the first, referred to as *global* and denoted by a G suffix, the values of L and U respectively are the minimum and maximum values of $w^*_{d,t}$ over the whole database. In the second, referred to as *local* and indicated by an L suffix, each document or query x is associated with its own L and U, which are the minimum and maximum among all of the values $w^*_{x,t}$ generated from x. That is, in the local transformation, a value $w^*_{x,t}$ is transformed with respect to the values of L and U of x – the document or query it belongs to.

Weighting scheme notation Two characters are used to denote a weighting scheme. The first character is a digit representing the indexing information level, from 1 to 5 (Table 1). The second is either G or L, indicating the transformation type.

3 Retrieval effectiveness

Ten runs were submitted for assessment, but an oversight meant that only nine were distinct. Five runs were for the topic distillation task, and four for the homepage finding task.

Homepage finding Our homepage finding runs are summarized in Table 2 and Figure 1. Indexing all components of the documents appears to be better than only indexing parts of each document. To some extent, this conclusion was expected. But the fact that method *1L* performs only slightly better than *3G* indicates that either our crude weighting scheme is contributing little, or that metadata, headings, and titles are relatively unimportant in determining the information content of documents.

When comparing MU609 with MU80A, it is perhaps surprising to see the role of indexing content for this task, and the not-so-good performance of anchor text. Before receiving the results we expected that the use of anchor text alone might give good effectiveness for this task, since anchor text represents a kind of "expert" external view of each document, while content provides a more subjective view. On the other hand, it is also possible that some of the pages sought in these topics were not the target of any other links in the collection.

On the other hand, run MU307 shows that using all metadata plus any anchor text is much worse than using anchor text alone. The use of subjective meta texts (description, keyword, title, heading) by homepage authors may be somewhat arbitrary.

Run label	Weighting scheme	Precision at 10
MU525	*1L*	0.1939
MU111	*2L*	0.1857
MU624	*3G*	0.1694
MU313	*4L*	0.1163
MU212	*4G*	0.1082

Table 3: Description of the topic distillation runs and their official results. For all runs, the title fields of the topics are used unchanged as input queries.

Another point drawn from Figure 1 is the small gaps in performance between the 5, 10 and 20 breakpoints on the number of documents retrieved. If our method is able to find the homepage at all, it appears to rank it highly.

Topic distillation While topic distillation is an interesting and practical task, we were unable to find an effective way of dealing with it using our weighting scheme. Our runs for this task are summarized in Table 3. Part of the poor performance in these runs is accounted for by our lack of use of the URL information that was provided, an omission that with hindsight is now obvious. The key problem in this task, we believe, is to locate good hubs and then discard any child pages connected to those hubs.

The performance of different weighting schemes has the same tendency as in the case of the homepage finding task. Scheme *1L* again perform best. In second place is scheme *2L*, which differs from *1L* only by excluding outgoing anchor text from indexing. Both are significantly better than using content text alone (scheme *3G*) which in turn is better than indexing only metadata and anchor text, or anchor text alone.

The results provide some support for the conjecture that a method that works well for one task can also work well for the other. And while there is no promise of high performance, it does provide hope that building a common system for both of the tasks might be possible.

4 System description and efficiency

Software The experiments were carried out using a modified version of *MG* (see `http://www.cs.mu.oz.au/mg/`). The main feature of *MG* for text retrieval is that it uses compression for document collections as well as for their indexes. This feature is especially appreciated when dealing with large collections like *.GOV*.

Changes have been made to *MG* to suit our needs, in both the indexing and the querying modules. While the changes are already reported in Anh and Moffat [2002], it is worth reiterating that the weighting scheme for document terms is integrated into the index, and that during query processing only a small amount of calculation is required.

Hardware A Beowulf cluster of 16 nodes was used for the experiments. Each of the nodes is an 800 MHz Intel Pentium III with 256 MB RAM, local hard disk of 40 GB, running Debian GNU Linux. The cluster is served by a 933 MHz Intel Pentium III with 1 GB RAM, a 9 GB SCSI disk for system needs, and four 36 GB SCSI disks in a RAID-5 configuration for data. There is a link with the capacity of 100 Mbits/second to and from each node as well as the server with a network switch.

Except for the server, our system uses all nodes for both indexing and querying. The server itself was employed only to deliver jobs to the nodes and to collect the final results. The indicative times reported below are for experiments in which there was no other activity on the hardware.

Data preparation Indexing and query processing was done in a distributed manner. The collection was split randomly across the nodes, with a separate index built for each subcollection. The intention was that each subcollection would be a homogeneous extract of the main collection, and that term frequencies and other collection statistics could be used locally within each subcollection without reference to the global values.

Each query was processed separately against each subcollection using the local index, and a global result listing compiled using the local scores. By assigning documents randomly to collections, we expected minimal degradation of retrieval effectiveness compared to a monolithic system.

Table 4 shows statistics for some of the subcollections and their indexes, built for the weighting scheme *1L*. The figures demonstrate that the subcollections are indeed almost indistinguishable, in terms of their gross statistics, and that the indexes are a small overhead. Their modest size is in part a function of the compression that is used, and in part a consequence of the removal of HTML tags during indexing. Note also that the total in the last row overstates the size of a comparable monolithic index, as there is considerable repetition within the sixteen separate vocabulary files.

Construction of indexes In the inverted list for a term t, each document d containing the term is associated with a quantized impact $\omega_{d,t}$ rather than a conventional $f_{d,t}$ value. Since the number of different quantized values is low, the index structure described by Anh et al. [2001] is appropriate. The pointers in each inverted list are partitioned

Subcollection	Subcollection statistics				Size of index	
	Size (MB)	Documents	Words	Pointers	MB	% of collection size
01	1,095	77,985	672,915	21,849,610	35.82	3.27
06	1,087	77,985	675,313	21,800,614	35.85	3.29
11	1,100	77,984	685,729	22,037,566	36.35	3.30
16	1,096	77,985	631,320	21,924,174	36.64	3.34
Over all subcollections:						
.GOV	17,469	1,247,753	n/a	350,770,758	579.59	3.31

Table 4: Statistics for four of the sixteen subcollections and their respective indexes, built on the 16-node Beowulf cluster for the *.GOV* collection. Document header fields were removed from documents and are not counted in the collection size. Anchor text is added to the corresponding destination document before building subcollections and indexes. Indexes are built for the weighting scheme *1L*, without any pruning, the maximum size among those tested. Sizes includes the cost of the inverted index and the vocabulary file. The last row of the table shows totals over all sixteen subcollections.

into blocks according to their quantized impact. Inside each block, documents are sorted in the increasing order of document numbers. The blocks themselves are arranged in decreasing impact order.

Building such an index requires only a small amount of additional computation compared with building a standard compressed index (described by Witten et al. [1999]). It took around 15 minutes to build the index for the *.GOV* collection and weighting scheme *1L*.

Query processing Query processing is also simple and fast [Anh and Moffat, 2002]. Note in particular that queries are evaluated in a distributed manner on the 16-node cluster. Just a few seconds suffice to run the 50 topic distillation queries and 150 homepage finding queries. The running time does not include time for retrieving the actual documents, and the task is presumed to be completed when the list of answer documents has been created.

5 Conclusion

Our participation in Web track has been limited by a number of simplifications. The link structure within the pages was not investigated, and the weighting scheme and retrieval strategy were not specifically adapted for the topic distillation task. The only feature added to the weighting scheme was the use of high weights for the appearances of words in anchor texts and certain tags. As a result, our results in the two 2002 tasks are not competitive with other systems. Effectiveness on the homepage finding task was better than for the topic distillation task, reflecting the fact that most of the changes to the system were made in favor of homepage finding.

On the other hand, the features we added to our system are not solely driven by this year's tasks, and involve elements that should also be added for the previous topic relevance task. Although the gain in retrieval effectiveness obtained to date is modest, the idea of building a common retrieval strategy for topic relevance, homepage finding and topic distillation seems to be possible.

It seems that a combined task – in which the search systems are not aware of the user's intention – would be a natural next step. That is, no preliminary information about the nature of a query is provided, and the system itself must decide whether it should it be regarded as homepage finding, information finding, or topic distillation. Systems could then approach the task in a number of possible ways, including using a sole system with queries treated equally, or automatically classifying queries into different categories for possible assigning a specific retrieval strategy.

Acknowledgement This work was supported by the Victorian Partnership for Advanced Computing.

References

V. N. Anh, O. de Kretser, and A. Moffat. Vector-space ranking with effective early termination. In W. B. Croft, D. J. Harper, D. H. Kraft, and J. Zobel, editors, *Proc. 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 35–42, New Orleans, LA, September 2001. ACM Press, New York.

V. N. Anh and A. Moffat. Impact transformation: Effective and efficient web retrieval. In M. Beaulieu, R. Baeza-Yates, S. H. Myaeng, and K. Järvelin, editors, *Proc. 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–10, Tampere, Finland, August 2002. ACM Press, New York.

A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In H.-P. Frei, D. Harman, P. Schäuble, and R. Wilkinson, editors, *Proc. 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29. ACM Press, New York, August 1996.

I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, San Francisco, second edition, 1999.

The University of Michigan at TREC2002: Question Answering and Novelty tracks

Hong Qi, Jahna Otterbacher, Adam Winkel, Dragomir R. Radev
University of Michigan
Ann Arbor, MI 48109
{hqi,jahna,winkela,radev}@umich.edu

November 3, 2002

1 Introduction

The University of Michigan participated in two evaluations this year. In the Question Answering Track, we entered three different versions of our system, NSIR, previously described in [1]. For the Novelty Track, we modified our multi-document summarizer, MEAD (`www.summarization.com/mead`) and submitted five runs with different input parameters.

2 Question Answering Track

We submitted three runs to the main task of the TREC Question Answering track this year. The system that we describe in this report is NSIR, a QA system being developed at the University of Michigan. NSIR uses a fine question taxonomy (2.1), extracts candidate answers along with nine features (2.2), and ranks the answers (2.3). We will show the architecture of the NSIR system (2.4). The official results from TREC will be provided, and we will also discuss how we plan to improve the performance of NSIR system (2.5).

2.1 Question Classification

To identify the semantic type of a question is a critical step when a QA system attempts to return phrasal answers. Our working assumption is that a finer taxonomy can better support answer extraction. We developed a hierarchical taxonomy that includes 141 different types. We believe that different types of questions should have different treatments. Therefore, we try to construct a tuned method to answer questions of each different type in the finer taxonomy.

By categorizing question "Where is the capital of Spain?" into *Captial* type instead of a more general type *Place*, for example, we could then use a predefined list of capital cities of each country to help answer this question. For another example, the target answer of a *Mountain* type question is likely to have the word "Mountain" or "Mt." followed by a mountain name which usually would be tagged as NNPs, i.e., proper nouns, by a typical part-of-speech tagger.

However, one drawback of using a finer taxonomy is that sometimes it is hard to classify a question into one single class. For example, the target answer of "Who won the Superbowl in 1982?" can be "San Francisco" which is a location or "49ers" which is the name of the team. Classifying each question into one single type may potentially lead to false negative cases, or, false positive answers, which is even worse. We adopted a probabilistic question classifier, which assigns normalized weights for each possible question type if the target answer could be in more than one categories. The probabilities of being each question type will be taken into account when ranking all the potential answers. Therefore, in our example, both "San Francisco" and "49ers" will be boosted if the question classifier assigns both "location" and "organization" as the expected types.

2.2 Answer Extraction

We obtain a list of phrases from the top relevant documents. Before we can rank these phrases, we compute the following features of each phrase.

Frequency How many times the phrase appears in the top documents.

Overlap How many words appears in both the phrase and the question.

Length NSIR defines a longest length for each expected type of answer. If the length of the phrase, i.e., the number of words that the phrase contains, is less than or equal to the predefined length, then it gets 1 on this feature; otherwise, it gets zero.

Proximity This feature reflects how close the phrase is to the content (non "stop words") words in the question. The more question words the phrase is close to, and the closer the phrase is to the question words, then the higher score the phrase can get.

POSSIG Part-of-speech signature. Some types of questions are expecting answers of certain part-of-speech tag sequence. For instance, the answers to a *Number* question are usually tagged as CDs (numeral, cardinal). POS signature shows whether or not the phrase matches the expected POS tags.

LEXSIG Lexical signature. Some questions expect certain words, symbols, or patterns to appear in the answers. For example, phrases containing "percent" or "%" are more likely to be the answer to a *Percentage* question.

Word List We build a local database for both closed and semi-open categories, such as country names, currencies, universities, etc. A candidate phrase will get a bonus if it is contained in the corresponding local database. Say for a *Language* type question, the phrases such as "Arabic", "Chinese", "English", "Latin", etc., will be boosted by this feature.

Named-entity NSIR uses named-entity taggers to locate named-entities in open categories. The BBN Identifinder [2] is used when the target answer is person names, locations or organizations. A time-related named-entity tagger has been developed for the questions of the types such as *Date*, *Festival*, *Time*, etc.

Web ranking This feature is motivated by the vast amount of data available on the web. Exploiting the redundancy of the web data has appeared since TREC 2001 Question Answering Track (e.g., [3, 4]). To get this feature computed, NSIR runs questions first on a web search engine such as Google, then download the top web documents, computes the above features, and combines all these features. The final score for each phrase becomes the "web ranking" feature for the same phrase that is extracted locally.

2.3 Answer Ranking

Once the features of the potential answer phrases have been computed, NSIR is ready to rank the list of phrases using the linear combination of the features. Note that the ways to linearly combine the features vary. Different question types have different weights for each feature. Table 1 shows the weights of features for three example question types, namely, *Author*, *Language*, and *Year* types. For example, as can be seen from the table, the *Named-entity* feature for *Language* type is 0 because no named-entity tagger is needed to answer *Language* type of questions. For another example, the *Web ranking* feature for *Year* type has lower weight than the other two types, which means that NSIR does not exploit the redundancy of web data for *Year* questions; this is because we are very likely to get recent years like "2002" which are not the intended answers of the TREC questions. Therefore, for each answer phrase, we can get a final score after combining the features in an appropriate way, then the phrase with the highest score becomes the top answer of the corresponding question. Until now, we have obtained *(qno, top answer, score)* for each question. The following part of this subsection will describe how we mark nil questions and rerank the answer list by confidence.

The corpus does not contain the answers to some questions, for which null responses will be marked as correct. Given that only one answer can be submitted for each question, returning a null response means that the top answer retrieved by our QA system will have to be given up. Our solution for marking nil questions is that any question whose top answer gets a score below a certain threshold, is marked as *NIL*. The threshold is learned by running NSIR on TREC-10 questions and observing a score so that the final scores of the NIL questions are below this score.

Our goal is to output the list of answers ranked by confidence. In our experiment, we use the final score of each top answer as the confidence level. Therefore, we rerank the answers by their final scores to get the intended ranking. However, for those NIL questions, we do need to find a way to boost their

Features	*Author*	*Language*	*Year*
Frequency	5	15	10
Overlap	-20	-20	-20
length	2	2	2
Proximity	15	5	5
POSSIG	3	3	0
LEXSIG	0	0	0
Word List	15	20	0
Named-entity	20	0	20
Web ranking	20	20	5

Table 1: Weights used in feature combination for question types *Author*, *Language*, and *Year*

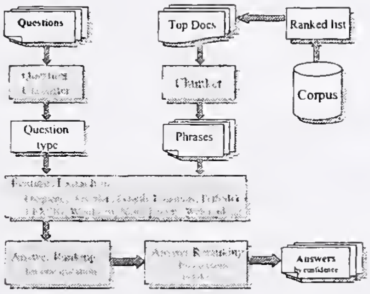


Figure 1: System Architecture of NSIR

rankings because they do not have a corresponding final score any more. Here we use a simple heuristic, namely, moving the first NIL question up to the 101st position, the second NIL question up to the 102nd position, and thereafter.

2.4 System Architecture

Figure 1 shows the architecture of the NSIR system. For each single question, we follow the steps below to extract the top answer.

a) Apply the question classifier to get the expected answer type (or types). This information will be used to compute the features for the candidate answers and rank the answers.

b) Obtain the top 20 documents according to the ranked list provided by

NIST; then apply LT-Chunk [5] on the top documents to get the phrases.

c) Once we get the expected answer type and the phrase list, we can start to compute the features for each phrase. Some features are independent of the expected answer type, such as *frequency*, *length*, etc. Other features are computed based on the answer type, such as *POSSIG*, *LEXSIG*, etc. To obtain the *Webranking* feature, NSIR sends the current question as a query to Google and ranks the phrases that appear in the web documents, then uses the final scores as the *Webranking* feature for the local phrases.

d) The steps above will produce a list of phrases with all the features computed. We linearly combine all the features with different weights predefined for the expected answer type. The candidate answer with the highest final score becomes the top answer to the question.

After NSIR gets the top answers along with their final scores for each question, it will apply to the answer list the heuristics for marking NIL questions and ranking answers by confidence that have been described in the last section. In the end, NSIR will output the answer list reranked according to the confidence levels.

2.5 Results and Discussion

We submitted three runs to this year's QA track. The main difference between them lies in the way of exploiting web data.

NSIRGN runs the questions on both local corpus and the web, using the results from the web as one more feature to rank the phrases that it gets from the local corpus.

NSIRG runs the questions only on the web, then locates the supporting local documents.

NSIRN runs the questions only on the local corpus without using the web.

Table 2 shows the official results on each of the three runs. NSIRGN outperformed the other two runs by about 6score, which suggests that the combination of local corpus and the web data performs better than using one of them alone.

In our experiment, we use the ranked list of documents provided by NIST, and run our system on the top 20 documents. Data shows that the top 20 documents contain answers for 311 out of the 500 questions. This is a major performance bottleneck for our system. More top documents should have been involved since the correct answers may be missed if the number of top documents is not enough. However, when we incorporate more top documents in the experiment, the number of candidate answers will be increased, which imposes difficulties on the later steps. Therefore, we believe that it is essential for a QA system to be able to rank high the right documents that support answering the input questions. If a large percentage of the top documents contain the correct

	NSIRGN	NSIRG	NSIRN
Wrong	379	388	404
Unsupported	18	16	7
Inexact	14	12	11
Right	. 89	84	78
Confidence-weighted Score	.283	.268	.269
# of NIL	175	195	63
Precision on NIL	.131	.128	.190
Recall on NIL	.500	.543	.261

Table 2: Official results on three runs of *NSIRGN*, *NSIRG*, and *NSIRN*

answers, then there will be much less noise when extracting answers, and much of the run time will also be saved.

One of the drawbacks in our system is that we did not effectively exploit the sentence level information. NSIR chunks the text into phrases right after it gets the top documents. Although the proximity feature carries some position information between the answer phrase and the question words, it is not able to distinguish between the cases where a phrase is always close to one of the question words and where a phrase appears near more question words. We plan to modify NSIR in the future so that it keeps the sentence level information when computing the proximity feature for the potential answer phrases.

There is another place that would improve the system performance. Recall the weights assigned to different features for each question type. Currently these weights are given manually by observing the system results on the previous TREC questions. Our question taxonomy contains so many classes, which makes it a hard task for humans to figure out the optimal weights for each question type. We plan to use machine learning techniques to learn these weights automatically.

This is our first time participating in the TREC QA evaluation (although the fourth author of this paper was earlier part of the IBM GuruQA project in TREC8 and TREC9 [6, ?]). Though we didn't get very exciting results, we gained valuable experience for our future work on using the Web to improve question answering. We will continue our research using the TREC framework to get continuous improvements.

3 Using a multidocument summarizer for detecting new and relevant information in the news domain

3.1 Introduction and approach

This year, the Michigan team submitted five runs to the Novelty Track competition. We have based our first novelty detection systems on an extractive summarizer, MEAD [7] [8], which is currently under development at the University of Michigan. Ideally, existing multi-document summarization systems should already be choosing sentences from a cluster of related documents that are both relevant and novel in terms of their information content. Therefore, we were curious as to how MEAD would perform in the evaluation with just a few modifications. Since the instructions given to the judges indicated that the list of novel sentences must be a subset of the relevant sentences, and since we had limited resources, we first devoted our energies to training MEAD to detect relevant sentences. In future evaluations, we plan to focus on novelty detection in its own right.

3.2 The MEAD summarizer

MEAD compresses a cluster of topically-related documents into a summary of the user's desired length. It uses sentence features, such as length and position in the source document in order to rank the sentences as to their perceived importance to the document cluster. A third feature used by MEAD is the centroid, which measures the extent to which each sentence contains a set of key words that are important to the overall cluster.

Next, the sentences are ranked according to their combined score, which is a linear combination of all the sentence features used. Finally, MEAD has a reranker, in which relationships between sentences can be represented, and used to change the sentence rankings. For example, the default version of the reranking script attempts to prevent redundancy. It uses a cosine similarity metric to compare each candidate sentence (for inclusion in the summary) to each higher-ranking sentence. If the candidate sentence is too similar in terms of lexical context, it is penalized and is not included in the summary. Finally, the top remaining n-percent of the sentences (with the compression rate n being determined by the user), are returned to the user as the summary. For the relevant and novel sentence detection tasks, we used MEAD to produce extracts, which specify the top n-percent of the ranked sentences.

3.3 Observations from the Sample Data

Before looking at the sample data, we first divided it into a training and development set. The first step in developing a strategy was to examine the judges' lists of relevant and novel sentences in the four training clusters. We also ran

Cluster	Judge	Set
303	A	train
303	B	train
359	A	train
379	A	train
379	D	development
423	C	development

Table 3: Training and development sets

MEAD on these four clusters, in order to examine the sentence features of the sentences selected by the judges.

We found that a large proportion (approximately 75% for both the new and relevant lists) of the chosen sentences had relatively high centroid scores (between 0.35 and 0.7) as compared to those sentences that were not chosen by the judges. In addition, we found that judges tended to select groups of sentences with high centroid scores. Judges were not likely to chosen a given sentence in isolation. In other words, it was very common to find many groups of consecutive sentences included in the judges' lists. One MEAD feature that we felt had little correlation to novelty detection or to sentence relevancy was the position score. Judges did not seem more likely to choose sentences that are close to the beginning of the document over those that are further away.

3.4 Modifications to the MEAD summarizer

Based on our observations of the judges selected sentences in the four training clusters, we felt that the centroid score should be weighted heavily in the sentence ranking algorithm. Additionally, we felt that we might be able to model the tendency of judges to choose groups of sentence with high centroid scores in the MEAD reranker. In our modified reranking algorithm, if two or more consecutive sentences have centroid scores that are greater than the average score for all sentences in the cluster, a bonus is added to their scores.

Tables 4 through 7 show the precision and recall of the default version of MEAD, and that of MEAD with the new reranker. Unless stated otherwise, the compression rate used was 10%, since typically, this rate yielded the best result. Note that the default version of MEAD has centroid and position weights of 1, such that the initial sentence scores are calculated as $score(sentence) = centroid * 1 + position * 1$. The length feature has a cutoff value of 9 words. If a sentence is less than 9 words long, it is thrown out.

The modified reranker helped in almost all of the new and relevant sets and across all four sample clusters, and in no case did it do worse than the default MEAD. Therefore, we decided to incorporate it into this year's systems.

The next step was to develop new MEAD sentence features using the query data provided for each cluster of documents. There was a title, description, and

System	Recall	Precision
MEAD (relevant)	0.375	0.146
MEAD + new reranker (relevant)	0.500	0.200
MEAD (new)	0.400	0.146
MEAD + new reranker (new)	0.467	0.171

Table 4: Training cluster 303A

System	Recall	Precision
MEAD (relevant)	0.227	0.122
MEAD + new reranker (relevant)	0.227	0.122
MEAD (new)	0.333	0.122
MEAD + new reranker (new)	0.333	0.122

Table 5: Training cluster 303B

System	Recall	Precision
MEAD (relevant)	0.200	0.004
MEAD + new reranker (relevant)	0.250	0.007
MEAD (new)	0.166	0.040
MEAD + new reranker (new)	0.222	0.053

Table 6: Training cluster 359A

System	Recall	Precision
MEAD (relevant)	0.137	0.091
MEAD + new reranker (relevant)	0.196	0.122
MEAD (new)	0.146	0.091
MEAD + new reranker (new)	0.208	0.130

Table 7: Training cluster 379A

System	Description	Official average P*R	Corrected average P*R
Umich1	Centroid 1, Position 1, Length 9	0.019	0.026
Umich2	Centroid 20, Position 1, Length 15	0.016	0.016
Umich3	Centroid 20, Position 1, Length 15, 7%	0.011	0.012
Umich4	Centroid 20, Position 1, Length 15, 13%	0.019	0.019
Umich5	Centroid 1, Position 1, Length 9 Query-title-word-overlap 1	0.034	0.042

Table 8: Umich systems and results for finding relevant sentences

System	Description	Official average P*R	Corrected average P*R
Umich1	Centroid 1, Position 1, Length 9	0.017	0.023
Umich2	Centroid 20, Position 1, Length 15	0.014	0.015
Umich3	Centroid 20, Position 1, Length 15, 7%	0.010	0.011
Umich4	Centroid 20, Position 1, Length 15, 13%	0.017	0.018
Umich5	Centroid 1, Position 1, Length 9 Query-title-word-overlap 1	0.033	0.039

Table 9: Umich systems and results for finding new sentences

narrative associated with each cluster. We used these queries in implementing a word overlap feature for each. For example, in calculating the query-title-word-overlap feature score for a given sentence, if the title had four words and the sentence contained two of the four words, its query-title-word-overlap feature score would be 0.5. In our experiments with these features, we found the title query to be most useful. This is because the title of the cluster typically had many key words that were important to it, while the narratives and descriptions tended to resemble instructions to the judges on how to choose sentences relative to the topic. Therefore, they usually contained several sentences and many non-content words, and so their word overlap features were not as helpful in choose new and relevant sentences.

3.5 The Michigan systems and results

Table 8 gives an overview of the five systems we submitted to this year's evaluation, as well as the systems' average performance using the precision*recall metric. It should be noted that during our runs on the test data, we encountered problems related to the conversion of the SGML documents to XML for use with MEAD. This resulted in our systems not being able to process all 50 test clusters. Therefore, in the table we will include both the official evaluation results as well as the corrected results for all 50 clusters.

Our best system was Umich5, which used as one of its features the query-title-word overlap measure. This system greatly outperformed our other four systems in detecting both relevant and new sentences. Judging by the preliminary results sent out by NIST, the average performance of Umich5 seems to be somewhere in the middle of all the systems submitted this year.

3.6 Conclusions

It being the first year of the Novelty Detection Track, as well as our first time attempting novelty detection per se, we have learned many things from participating in the evaluation. One lesson learned about our back-end summarizer, MEAD, is that we must continue to make it more robust to the format of its text input files. As briefly mentioned previously, we encountered some problems with the conversion from SGML to XML during the testing that we did not see during our runs with the training data. We have since addressed this issue in our text cleaning scripts within MEAD.

We introduced a new feature within MEAD, query-title-word-overlap, which proved to be useful in system Umich5. However, we found that sentence overlap with the other query attributes, narrative and description of the clusters of documents, were not as helpful. We suspect this is because they appear to be more or less instructions to the judges about what to consider when choosing new and relevant sentences for the given cluster, and since they are longer than the titles and contain more function words, their overlap features do not really measure the extent to which a given sentence contains lexical items that are important to the overall cluster.

We have developed a new reranker within MEAD, which boosts the scores of groups of adjacent sentences that have large centroid scores, relative to the average centroid score for all of the sentences in the given cluster of documents. During our experiments with the training data, this seemed to model the tendency of judges to choose groups of consecutive sentences, rather than those in isolation. This reranker increased our precision and recall for both the new and relevant sentences sets for three out of four of the sample data clusters and left the fourth cluster's scores unchanged, when all other MEAD parameters were set at the default values. We might also try looking at using the query-title-word-overlap feature in this new reranker. However, in cases where the title is only a few words long, this might not work as well in attempting to boost scores of adjacent sentences, since one would expect authors to avoid using the same words in nearby sentences.

As is always the case, the judges' choices that we studied in the sample data while building our system seemed idiosyncratic at times, such that we could never obtain similar scores for multiple judges on a given cluster of documents. For example, for training cluster 303, we consistently got much higher precision and recall with judge A than with judge B. Therefore, in our training and experiments for next year's competition, we should use many different metrics in our self-evaluations, such as the intersection, union and mininum agreement of judges' new and relevant files.

Finally, for next year's competition, we will work on developing different strategies for detecting new versus relevant sentences. The new sentences are always a subset of the relevant ones. Therefore, we need to identify features that can distinguish the redundant sentences in the relevant lists that the judges are selecting out in order to create the file of new sentences. In short, this year we have seen how our summarizer, MEAD, performed on the novelty tasks with just a few simple modifications. Hopefully, by the next competition we will have identified more sophisticated features that we might implement, in order to enhance our system's performance.

References

[1] Dragomir R. Radev, Weiguo Fan, Hong Qi, Harris Wu, and Amardeep Grewal. Probabilistic question answering from the web. In *The Eleventh International World Wide Web Conference*, Honolulu, Hawaii, May 2002.

[2] Daniel M. Bikel, Richard L. Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211-231, 1999.

[3] Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. Data-intensive question answering. In *Proceedings of the Text Retrieval Conferences*, Nov 2001.

[4] C. L. A. Clarke, G. V.Cormack, T. R. Lynam, C. M. Li, and G. L. McLearn. Web reinforced question answering (multitext experiments for trec2001). In *Proceedings of the Text Retrieval Conferences*, Nov 2001.

[5] Language Technology Group. Lt chunk software. http://www.ltg.ed.ac.uk/software/chunk/.

[6] John Prager, Eric Brown, Anni Coden, and Dragomir Radev. Question-answering by predictive annotation. In *Proceedings, 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Athens, Greece, July 2000.

[7] Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. Centroid-based summarization of multiple documents: sentence extraction, utility-based evaluation, and user studies. In *ANLP/NAACL Workshop on Summarization*, Seattle, WA, April 2000.

[8] Dragomir Radev, Simone Teufel, Horacio Saggion, Wai Lam, John Blitzer, Arda Çelebi, Hong Qi, Elliott Drabek, , and Danyu Liu. Evaluation of text summarization in a cross-lingual information retrieval framework. Technical report, Center for Language and Speech Processing, Johns Hopkins University, Baltimore, MD, June 2002.

The QUANTUM Question Answering System at TREC-11

Luc Plamondon Guy Lapalme
RALI/DIRO, Université de Montréal
CP 6128, Succ. Centre-Ville
Montréal, Québec, Canada, H3C 3J7
{plamondl, lapalme}@iro.umontreal.ca

Leila Kosseim
Concordia University
1455 de Maisonneuve Blvd. West
Montréal, Québec, Canada, H3G 1M8
kosseim@cs.concordia.ca

Abstract

This year, we participated to the Question Answering task for the second time with the QUANTUM system. We entered 2 runs for the main task (one using the web, the other without) and 1 run for the list task (without the web). We essentially built on last year's experience to enhance the system. The architecture of QUANTUM is mainly the same as last year: it uses patterns that rely on shallow parsing techniques and regular expressions to analyze the question and then select the most appropriate extraction function. This extraction function is then applied to one-paragraph long passages retrieved by Okapi to extract and score candidate answers. Among the novelties we added to QUANTUM this year is a web module that finds exact answers using high-precision reformulation of the question to anticipate the expected context of the answer.

1 Introduction

TREC-11 marks the second year of existence of QUANTUM, the QUestion ANswering Tool of the University of Montreal. As for last year, we used QUANTUM to participate to the main and the list task, and we did not enter the context task. This year's version of QUANTUM is similar in essence to last year's version, but we enhanced specific modules that provided poor performances last year and we added a module to search for exact answers on the web. Following the conclusions we came to last year [1, 2], we decided to drop our own information retrieval system to rely solely on Okapi[1] [3] since the latter led to clearly better results. Also, we fine-tuned our answer extraction functions by introducing weights in the computation of answer scores.

The TREC-10 version of QUANTUM is described in [1] and an analysis of the results is presented in [2]. We summarize some sections that are still relevant to the understanding of this year's version, while we delve into the new features in more detail.

2 Components of Questions and Answers

QUANTUM uses the same technique for question analysis as last year [2]. To see how we decompose a question, let us consider question *#302 – How many people die from snakebite poisoning in the US per year?* and its answer. As shown in Fig. 1, the question is decomposed in three parts: a *question word*, a *focus* and a *discriminant*, and the answer has two parts: a *candidate* and a variant of the question *discriminant*.

The *focus* is the word or noun phrase that influences our mechanisms for the extraction of candidate answers (whereas the *discriminant*, as we shall see in Sect. 4.4, influences only the scoring of candidate answers once they are extracted). The identification of the focus depends on the selected extraction mechanism; thus, we determine the focus with the syntactic patterns we use during question analysis. Intuitively, the focus is what the question is about, but we may not need to identify one in every question if the chosen mechanism for answer extraction does not require

[1] Okapi-Pack: www.soi.city.ac.uk/~andym/OKAPI-PACK

$$\textbf{Q:}\quad \underbrace{\textit{How many}}_{\text{question word}} \quad \underbrace{\textit{people}}_{\text{focus}} \quad \underbrace{\textit{die from snakebite poisoning in the U.S. per year?}}_{\text{discriminant}}$$

$$\textbf{A:}\quad \underbrace{\textit{About 10 people}}_{\text{candidate}} \quad \underbrace{\textit{die a year from snakebites in the United States.}}_{\text{variant of question discriminant}}$$

Figure 1: Example of question and answer decomposition. The question is from TREC-9 (# 302) and the answer is from the TREC document collection (document LA082390-0001).

it.

The *discriminant* is the remaining part of a question when we remove the question word and the focus. It contains the information needed to pick the right candidate amongst all. It is less strongly bound to the answer than the focus is: pieces of information that make up the question discriminant could be scattered over the entire paragraph in which the answer appears, or even over the entire document. In simple cases, the information is found as is; in other cases, it must be inferred from the context or from world-knowledge.

We use the term *candidate* to refer to a word or a small group of words, from the document collection, that the system considers as a potential answer to the question. Candidates are usually noun phrases (see Sect. 4.3.1 for a discussion on exact answers).

3 Runs Submitted at TREC-11

This year, we participated to the main task and the list task. We developed two versions of QUANTUM for the main task: one version that does not make use of the web, and one that does. For the list task, only the no-web version was ready by the time of the competition. So only one run was submitted for the list task and two for the main task.

4 Architecture of the Core System

The current version is a continuation of last year's system but with enhancements suggested in our analysis of last year's results and with some changes required by the new specifications of the 11th edition of TREC (exact answer. single answers, etc.). After analyzing last year's performance, we concentrated our efforts in improving the question analysis module to correctly analyze more questions and the scoring module to better score the candidates. In particular, we wanted to better weight the contribution of WordNet and the named-entity tagger depending on the type of question. Last year, our strategy to insert NIL answers in our candidate set dropped our score significantly (from 0.223 to 0.199). We believe that our strategy was sound, but our scoring of the candidates was such that we would have been better off without inserting NIL answers. This is why, this year, we did not attempt to insert NIL answers.

Let us step through the 4 basic steps of the system: question analysis, passage retrieval and tagging, candidate extraction and candidate scoring.

4.1 Question Analysis

To analyze a question, we use a tokenizer, a part-of-speech tagger and a noun-phrase chunker (NP-chunker). QUANTUM then applies a set of handmade patterns based on words, on part-of-speech tags and on noun-phrase tags, in order to select the most appropriate function for answer extraction. Last year, only 40 such patterns were used, but they correctly classified 88 % of the 492 TREC-10 questions. To increase the performance of the classification module, we added 20 more patterns to account for last years mistakes and new question formulations. Once the question is classified, an extraction function determines what criteria a group of words from a document should satisfy to constitute a valid answer candidate; for example, a *location* should begin with a capital letter while a *measure* should include a number and a measure unit. We also added a *synonym* extraction function to our last year's set of 11 functions (Table 1) because some questions in the TREC-10 corpus required it. Extraction functions such as *definition* were less useful this year because the questions were mainly named-entity targeted.

Like last year, each function triggers a search mechanism to identify candidates in a passage based on the passage's syntactic structure or the semantic relations of its component noun phrases with the question focus. More formally, we have $\mathcal{C} = f(\rho, \varphi)$, where f is the extraction function, ρ is a passage, φ is the question focus and $\mathcal{C}$ is the list of candidates found in ρ. Each element of $\mathcal{C}$ is a tuple (c_i, d_i, s_i), where c_i is the candidate, d_i is the number of the document containing c_i, and s_i is the score assigned by the extraction function.

4.2 Passage Retrieval and Tagging

The extraction of candidates is a time-consuming task. Therefore, we select the shortest, albeit most relevant, passages of the document collection before we begin answer extraction. To do so, we use the Okapi system to retrieve variable-length passages. Passage retrieval has not changed from last year; however, experiments showed that our own fixed-window IR did not achieve as good results as Okapi. So this year, we exclusively used Okapi. Okapi is an information retrieval engine that has the ability to return relevant paragraphs instead of whole documents [3]. We feed it with the question as a query and we set it up so that it returns 30 one-paragraph-long passages (the average length of a passage, or paragraph, is 350 characters).

Since the answers to the TREC-11 factual questions are usually short noun-phrases (Sect. 4.3.1), we run our NP-chunker on the most relevant passages. Our chunker looks for specific sequences of part-of-speech tags, which are given by our tagger. In addition, we run a named entity extractor on the passages because the candidates sought by extraction functions such as $person(\rho)$, $time(\rho)$ and $location(\rho)$, are named entities. For this step, we use the Alembic Workbench system developed at Mitre Corporation for the Message Understanding Conferences (MUC). Amongst all the named entity types that Alembic can recognize [4], we currently use only PERSON, ORGANIZATION, LOCATION, DATE and TIME entities.

The tagged passages are then passed to the previously selected extraction function to identify and score candidate answers.

4.3 Extraction of Candidates

Given the extraction function f chosen after question analysis, the question focus φ and a set of tagged passages ρ_j, candidates c_i are extracted along with their document number d_i and their score s_i (Sect. 4.1). To do so, each function is implemented by a set of search strategies that involve words, part-of-speech tags, semantic relations (mainly hypernym/hyponym relations given by WordNet) and named entities identified by Alembic. Table 1 presented earlier shows some examples of what extraction functions look for. During the extraction phase, we seek a high recall rate, no matter whether candidates are cited in a context that matches the question discriminant; we shall use a combination of scores to account for the context later.

4.3.1 Answer Exactness

A major difference with the TREC-10 QA track is that answers must be exact. This means there is no limit imposed on the length of an answer string, but the string must not contain an incomplete answer nor must it contain more information than requested by the question. What is considered "too much" or "not enough" was not clearly defined at the time the competition was held; the problem was left to the assessors' judgment. Another consideration for having exact answers, although not relevant to TREC, is that potential users of QA systems are more likely to find *the red, green and white flag* more pleasant to read than *), that the red, green and white flag of the Ital.*

We decided to keep the strategy we employed last year, which is to extract all the noun-phrases in a retrieved passage and then to test, using the selected extraction function, whether each of them is an interesting candidate. That means that a candidate is always at least a complete NP and it never encompasses more than one NP, except in a limited number of cases that we explicitly foresaw when writing the extraction functions. We are aware that this is not suitable to all questions but we believe it is a simple way to achieve a satisfying level of answer exactness most of the time.

There are cases where a NP is not long enough (our definition of a single NP does not include conjunctions of embedded NP), typically when (1) a question seeks more than one entity: *#1422 – What two European countries are connected by the St. Gotthard Tunnel under the Alps?* A: *Switzerland and Italy*[2] and (2) the answer is a title or a quote: *#1832 – What did Walter Cronkite say at the end of every show?* A: *"and that's the way it was"*. There are also cases where a NP can be too long. For example, *Louise Veronica Ciccone* would be an inexact answer to question *#1723 – What is Madonna's last name?* A: *Ciccone*.

After examining answer patterns for TREC-8, TREC-9 and TREC-10 questions, we estimated that

[2] Even though this question has a list-task style, it is part of the main task.

Function	Example of question and sample of answer patterns
definition(ρ, φ)	Q: *#897 – What is an atom?* ($\varphi = atom$) A: <hypernym of *atom*>, <*atom* or hyponym of *atom*> A: <*atom* or hyponym of *atom*> (<hypernym of *atom*>) A: <*atom* or hyponym of *atom*> is <hypernym of *atom*>
specialization(ρ, φ)	Q: *#1684 – What card game uses only 48 cards?* ($\varphi = card\ game$) A: <hyponym of *card game*>
cardinality(ρ, φ)	Q: *#1761 – How many black keys are on the piano?* ($\varphi = black\ keys$) A: <number> <*black keys* or hyponym of *black key*>
measure(ρ, φ)	Q: *#1715 – How much vitamin C should you take in a day?* ($\varphi = vitamin\ C$) A: <number> <hyponym of *unit*> of <*vitamin C* or hyponym of *vitamin C*>
attribute(ρ, φ)	Q: *#1420 – How high is Mount Kinabalu?* ($\varphi = high$) A: Various patterns
synonym(ρ, φ)	Q: *#1651 – What is another name for the North Star?* A: *the North Star*, also known as <NP> A: <NP>, also known as *the North Star*
person(ρ)	Q: *#1424 – Who won the Oscar for best actor in 1970?* A: <PERSON named entity>
time(ρ)	Q: *#1676 – When was water found on Mars?* A: <TIME named entity> A: <hyponym of *time_period*>
location(ρ)	Q: *#1483 – Where is the highest point on earth?* A: <LOCATION named entity>
manner(ρ)	Q: *#1446 – How did Mahatma Gandhi die?* A: Not implemented for TREC
reason(ρ)	Q: *#902 – Why does the moon turn orange?* A: Not implemented for TREC
object(ρ)	Default function A: <NP>

Table 1: Extraction functions, examples of TREC questions and samples of answer patterns. Hypernyms and hyponyms are obtained using WordNet, named entities are obtained using Alembic and NP tags are obtained using an NP-chunker. When we mention the focus in an answer pattern, we also imply other close variants or a larger NP headed by the focus.

only 2 % of the questions could not be answered by a NP. Therefore, we decided to make no significant change to our system regarding answer exactness.

4.4 Scoring of Candidates

The final score of a candidate is computed as in Eq. 1:

$$score = \alpha \cdot extraction\ score + (1 - \alpha) \cdot passage\ score \quad (1)$$

with α, the *extraction score* and the *passage score* ranging from 0 to 1 (we describe the *extraction score* and the *passage score* below).

We dropped the *proximity score* term that we used last year. This 3rd term was meant to favor candidates that were surrounded by question keywords but it did not have a noticeable influence in the tests we conducted this year.

We found the optimal value $\alpha = 0.75$ by maximizing the performance of QUANTUM on a subset of TREC-8, TREC-9 and TREC-10 questions. We discarded no-answer questions and questions that QUANTUM does not analyze correctly.

The final scores given by Eq. 1 range from 0 to 1. Last year, scores awarded by QUANTUM did not have an upper bound, so candidates extracted using different extraction functions were hardly comparable. The comparability of scores seemed a prerequisite for detecting no-answer questions using a threshold (Sect. 4.5) and for a final re-ordering of the questions based on confidence (Sect. 6).

4.4.1 The Extraction Score

The *extraction score* measures how much a candidate satisfies various criteria that all valid candidates should meet. The set of criteria is specific to each type of question, thus to each extraction function. Criteria are weighted and a candidate does not have to satisfy all of them. Eq. 2 shows the criteria for the *time* function:

$$\text{score}_{time} = \max(entity, hyponym) \cdot penalty \quad (2)$$

with $entity = 0.5$ if the candidate has been tagged as a TIME named entity by Alembic ($entity = 0$ otherwise), $hyponym = 0.25$ if the candidate is a hyponym of the WordNet synset *time_period* or of another selected synset ($hyponym = 0$ otherwise), and $penalty = 0.75$ if the candidate contains one of the question keywords ($penalty = 1$ otherwise so that *score* is not reduced).

The values of the parameters for the 12 extraction functions were found by maximizing the performance of QUANTUM on the same question subset as described above. An outcome of the optimization is the reduction of the influence of WordNet to the benefit of Alembic for the *person* function (WordNet: 0, Alembic: 1), for the *location* function (WordNet: 0, Alembic: 1) and for the *time* function (WordNet: 0.25, Alembic: 0.75). This was expected because our analysis of last year's results led us to the conclusion that WordNet was rather a source of noise when a named entity extractor could be used instead.

4.4.2 The Passage Score

While the extraction score is concerned only with the form and type of a candidate, the *passage score* attempts to take into account the supplemental information brought by the question discriminant. It measures how confident we are in the passage where the candidate is found. For this measure, we use the score given to the passage during its retrieval by Okapi. However, this year, we normalize the score of each passage over the score of the best-scoring passage to have a *passage score* — and thus a final score (Eq. 1) — between 0 and 1. Since the question discriminant is likely to appear in the text under a slightly different form and to be scattered over several sentences around the sought candidate, we believe that an IR engine is the best tool for measuring the concentration of elements from the discriminant in a given passage.

4.5 No-Answer Questions

At TREC-10, we measured the score drop between the ranked candidates for a question and we inserted a NIL answer when the score drop was higher than a threshold. This meant the system would rather say there is no suitable answer in the document collection than propose any candidate at a worse rank. We had chosen to use a threshold on normalized score drops instead of a threshold on absolute scores because different extraction functions would use different score scales. The major drawback of that method is the impossibility to propose a NIL answer at first rank (unless QUANTUM finds no candidate at all, which seldom happens because the extraction functions are very permissive).

This year, we attempted to use scoring methods that produce comparable final scores no matter how the candidates are extracted. Our goal was to set a threshold on the final score below which a NIL answer would be preferred. Unfortunately, because of the small number of criteria used when computing the extraction score and because of their boolean behaviour, the scores of all the candidates tend to cluster around a few values. We observed that the scores of first-rank candidates ranged from 0.5 to 1, with about one third of them at 0.5. Thus, a NIL insertion based on a score threshold would have resulted in a NIL answer for 33 % of the TREC-10 questions, which seemed too far from reality (10 %) to be an improvement to the system. Therefore, we decided not to detect no-answer questions.

5 Architecture of the Web Module

Following several systems from last year, and because TREC-11 questions are of general domain, the web proved to be an interesting source of answers [5, 6]. We present here how a new module of QUANTUM makes a simple use of the web to retrieve exact answers.

Our use of the web differs from most of last year's participants. To use the web, two strategies could be used: either aiming at high recall or high precision. High recall would mean retrieving a large list of candidate answers and using a good scoring strategy to rank the candidates. This is similar in essence to answer redundancy. We decided to go for the other approach: high precision of the answers at the expense of recall.

We do not seek supplemental documents from the web to complement the set of documents retrieved from the TREC collection, because QUANTUM has enough answer candidates from the TREC collection. The problem is to correctly identify the answer from the list of candidates and score it such that it is positioned at the top of the list. So we use the web to retrieve candidates only through high-precision methods in such a way that the web does not return an answer often, but when it does, the answer is expected to be the correct one. To do so, we look on the web for an exact sentence that could naturally be the formulation of the answer to the question.

To formulate an answer pattern, the TREC question is turned into its declarative form using a set of hand-made patterns. For instance, *#1697 – Where is the Statue of Liberty?* is reformulated as "`the Statue of Liberty is <LOCATION>`". We call this reformulation the *expected context of answer* because we expect to find this pattern, with the correct answer in place of `<LOCATION>`, at least once on the web.

We then use *Yahoo!* to search for web pages that contain the known part of the expected context (here, the phrase "`the Statue of Liberty is`") and we identify answer candidates by unification. We do simple validity checks on candidates, such as testing the length of a candidate or whether a candidate for a location begins with a capital letter, but as for now the tests are not as sophisticated as the extraction functions displayed in Table 1.

The web module can identify about 10 general types of answers (*eg.* `<LOCATION>`, `<NUMBER>`, `<CLAUSE>`, ...). However, by using a conjunction of expected contexts, we can impose more constraints on a candidate. For example, the question *#1851 – Which country colonized Hong Kong?* is reformulated as "`<CLAUSE> colonized Hong Kong`", where `<CLAUSE>` can be any string. To further restrict the candidate, we impose a second context for the candidate to satisfy: "`<CLAUSE> is a country`". The search thus becomes a conjunction of the two strings "`<CLAUSE> colonized Hong Kong`" and "`<CLAUSE> is a country`" (they can be found in separate web pages).

We score the candidates once they are extracted. A candidate that passes the validity checks starts with a score of 0.65. For each additional occurrence of the candidate found in any of the expected contexts derived from the question, the difference to one is divided by 2, thus raising the score to 0.825, then 0.9125 and so on. A candidate that does not meet the validity criteria but appears where an answer was expected receives a score of 0.1. Each additional occurrence of the candidate boosts the score by 0.1, as long as the resulting score does not exceed 0.6. Therefore, a presumably invalid candidate never has a higher score than a presumably valid candidate.

After we find an answer on the web, we use Okapi to perform a search in the TREC-11 document collection in order to have a document number to accompany the candidate. For the questions where either no candidate is found on the web or none of them can be matched with a document number, QUANTUM proceeds without the web module, as described in Sect. 4.

To test the performance of the web module, we conducted experiments with TREC-9 and TREC-10 questions. In the TREC-9 and TREC-10 document collection, we seldom find an expected context (we do for only 10 % of the questions). However, when we search on the web, we find at least one occurrence of an expected context for 43 % of the questions. Of these, the answer identified by unification is correct 51 % of the time, and 45 % of them appear at first rank. In clear, 10 % of the TREC questions are cor-

Run name	# right	Score	Max	Min
UdeMmainNoW	37	0.080	0.266	0.003
UdeMmainWeb	29	0.057	0.222	0.001

Table 2: Confidence-weighted scores of the 2 main-task runs compared to the maximum and minimum possible scores given the number of correct answers.

rectly answered only by searching for expected contexts of answers on the web and by performing basic validity checks.

We also tried to run QUANTUM with and without the web module, and then to keep the best-scoring candidate of either version. Results were worse than with the web module alone, probably because scores obtained with expected contexts and with Eq. 1 should be weighted to be appropriately comparable.

6 Confidence

Answers for the main task have to be submitted to NIST in decreasing order of confidence. To do so, we simply sort them according to their score. Table 2 shows how the sorting affected our runs by comparing their confidence-weighted score to a hypothetical maximum score (all correct answers at the top of the sorted list) and minimum score (all correct answers at the end of the sorted list), given the number of correct answers. Our ordering could be improved considerably.

7 List task

The only supplemental difficulty of the list task is the identification of the desired number of items specified in the question. We had to adapt the new question analysis patterns we added this year in the same way that we adapted last year's main task patterns for the list task. Once the system has analyzed a list question with the appropriate set of patterns, it proceeds exactly as for the main task (unfortunately, only the no-web version described in Sect. 4 was ready for TREC-11), except that it keeps the desired number of candidates instead of only the best one. For simplicity, duplicate candidates are eliminated even if they come from different documents (they might not be real duplicates because one candidate might be supported by its document and the other not).

8 Discussion

When we conducted our pre-competition tests on the TREC-10 question corpus, we estimated that the no-web version of QUANTUM could correctly answer 14 % of the questions, and that the web version would perform even better with a correct answer for 17 % of the questions. This was an improvement over last year's version of QUANTUM, which correctly answered 12 % of the questions (this data comes from last year's best run — with 50-character answers, but keeping only 1 answer per question). However, our estimations were made using an automated evaluation procedure that could not detect unsupported answers nor inexact ones (the *lenient* evaluation).

Table 3 shows the official evaluation details of our 2 main-task runs. We will exclude the effect of confidence weighting from our analysis and we will use Table 4 to compare a strict evaluation (the percentage of right answers) against a lenient evaluation that includes inexact and unsupported answers (as an automatic evaluation script would do).

The lenient evaluation in Table 4 is closer to our pre-competition estimations. The web module is clearly an improvement to QUANTUM (15 % of correct answers when using the web and 9 % without). However, because of the important number of web answers that are unsupported (40 unsupported answers when using the web, 2 without), a strict evaluation cuts the performance by half, making the web version even worse than the no-web version (6 % of correct answers when using the web, 7 % without).

As for answer exactness, the proportion of exact answers over all the supported answers is about 16 % for both runs. We consider this a satisfactory result given the simplicity of the method we have chosen (single NPs only) and given that some of the errors are tagging errors, thus not related to the method itself.

The list-task run achieved an accuracy of 0.07, while our last year's best run achieved an accuracy of 0.15. Since all the answers that QUANTUM found this year are distinct and very few are unsupported, we attribute the performance decrease to the additional constraints on answer length (candidates are much smaller than 50 characters and some are inexact).

9 Conclusion

Even though we fine-tuned last year's version of QUANTUM by adding more question analysis patterns, by introducing function-specific weights in the

Run name	Web	# wrong	# unsupp'd	# inexact	# right	CWS
UdeMmainNoW	no	454	2	7	37	0.080
UdeMmainWeb	yes	425	40	6	29	0.057

Table 3: Detailed evaluation of the 2 main-task runs with their confidence-weighted score (CWS).

Run name	Web	Strict	Lenient
UdeMmainNoW	no	7 %	9 %
UdeMmainWeb	yes	6 %	15 %

Table 4: Strict (right answers only) and lenient (right, inexact and unsupported answers) evaluations of the 2 main-task runs. The score is the ratio over 500 questions.

computation of the candidates' scores and by decreasing the influence of WordNet to the benefit of Alembic, the increasing difficulty of the task resulted in a lower performance for this year. We feel that answer exactness can be achieved reasonably well by retrieving single-NP answers only, but candidate scoring, document support and no-answer questions are still challenging issues.

For the first time, we used the web as a source of answers. We derived from the questions the context in which we expected the answers to appear. Not taking into account answer exactness and document suppport, we found that 10 % of TREC-style questions can be correctly answered this way. We plan to go further in this direction by improving the generation of expected contexts and the validation of the candidates' semantic types.

Acknowledgments

We wish to thank Louis-Julien Guillemette whose programming skills gave QUANTUM access to the web. We also wish to express our greatest appreciation to the technical support staff of the DIRO department of the U. de Montréal. As Murphy's law would have predicted, the lab's file server crashed during the week of the competition. Without their fast and skillful response, we could not have made it on time this year.

This project was financially supported by the Bell University Laboratories (BUL) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] L. Plamondon and L. Kosseim. QUANTUM: A Function-Based Question Answering System. In *Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence (AI 2002)*, pages 281–292, Calgary, Canada, 2002.

[2] L. Plamondon, G. Lapalme, and L. Kosseim. The QUANTUM Question Answering System. In *Proceedings of The Tenth Text Retrieval Conference (TREC-X)*, pages 157–165, Gaithersburg, Maryland, 2001.

[3] S.E. Robertson and S. Walker. Okapi/Keenbow at TREC-8. In *Proceedings of TREC-8*, pages 151–162, Gaithersburg, Maryland, 1998.

[4] J. Aberdeen, J. Burger, D. Day, L. Hirschman, P. Robinson, and M. Vilain. MITRE: Description of the Alembic System as used for MUC-6. In *Proceedings of the Sixth Message Understanding Conference*, San Francisco, California, 1995. Morgan Kaufman Publishers.

[5] C.L.A. Clarke, G.V. Cormack, T.R. Lynam, C.M. Li, and G.L. McLearn. Web Reinforced Question Answering (MultiText Experiments for TREC 2001). In *Proceedings of The Tenth Text Retrieval Conference (TREC-X)*, pages 673–679, Gaithersburg, Maryland, 2001.

[6] E. Brill, J. Lin, M. Banko, S. Dumais, and A. Ng. Data-Intensive Question Answering. In *Proceedings of The Tenth Text Retrieval Conference (TREC-X)*, pages 393–400, Gaithersburg, Maryland, 2001.

Progress in General-Purpose IR Software

Gregory B. Newby
UNC Chapel Hill

Abstract

TREC 2002 experiments were run, but not submitted in time for inclusion in the official conference results. Post-hoc analysis is included in this paper. Progress on general-purpose IR software for experimental use has been very good, and some features of the software are described. A new focus on IR for grid computing, GridIR, is described.

Introduction

The IRTools software developed by the author and his colleagues was used again this year. VSM's Lnu.Ltc and LSI retrieval models were used. Options for automatic query expansion and pseudo relevance feedback were available, as well as a variety of components for stoplist processing etc.

Unfortunately, runs were completed just a few hours too late and so were not included in the TREC conference results. Interactive track data collection is not yet completed, but the research design is presented below.

Completed runs for TREC 2002 included:

CLIR: Monolingual Arabic Web: Topic Distillation

Software Overview

IRTools is intended to be a general-purpose toolkit for information retrieval research. It was funded in part by the NSF through an Information Technology Research grant. The source code for IRTools is available at `http://sourceforge.net/projects/irtools`.

In early 2003, IRTools is nearing readiness for use by other IR researchers. It offers high-performance indexing and retrieval, and many of the features found in other experimental IR systems – but with more of an emphasis on allowing the programmer to change parameters, extend functionality, etc. Completion of IRTools for public release is scheduled for May 2003. At that time, modules to be included are:

- Indexing for multiple document type: XML, text and HTML
- Processing for English, Arabic, Chinese and other languages
- Local file indexing as well as remote harvesting
- Several fundamental IR techniques:
    - Enhanced Boolean
    - VSM
    - LSI
    - Information space

- Several fundamental IR enhancements:
  - Query expansion
  - Document summarization

The toolkit uses the BerkeleyDB for the back end database and Michael Berry's SVDPACKC for eigensystems. Other components are home-grown. The system runs on Unix and Linux systems with the GCC compiler and has been tested extensively on Linux and Solaris systems.

CLIR Arabic Monolingual Results

Two monolingual Arabic runs were completed. One utilized the entire document; the other utilized the title only (intended for high early precision). Basic tools provided by the track coordinators were applied to modify the topic character set to match the document set, but no other processing was done (i.e., no stemming, stopwords, or analysis of document structure). This "bag of words" approach was envisioned as a starting point for further experimentation.

For this run, the VSM was used with Lnu.Ltc weighting (pivoted document length normalization with the cosine measure of association).

Note that in the title only run, all other sections of each document were ignored. Indexing for both title only and the whole document ran as part of one IRTools indexing program and took about an hour for the 895MB of text (383K documents with 660K unique terms). Summary results are presented in Tables 1 and 2.

Table 1: Monolingual Arabic for irtta (title only)

```
      Total number of documents
over all queries
          Retrieved:       335
          Relevant:       3055
          Rel_ret:         121
      Interpolated Recall -
Precision Averages:
          at 0.00         0.5461
          at 0.10         0.1937
          at 0.20         0.0122
          at 0.30         0.0115
          at 0.40         0.0000
          at 0.50         0.0000
          at 0.60         0.0000
          at 0.70         0.0000
          at 0.80         0.0000
          at 0.90         0.0000
          at 1.00         0.0000
```

```
      Average precision (non-
interpolated) for all rel
docs(averaged over queries)
                         0.0352
      Precision:
        At    5 docs:    0.2889
        At   10 docs:    0.2111
        At   15 docs:    0.1741
        At   20 docs:    0.1611
        At   30 docs:    0.1333
        At  100 docs:    0.0667
        At  200 docs:    0.0336
        At  500 docs:    0.0134
        At 1000 docs:    0.0067
      R-Precision (precision after
R (= num_rel for a query) docs
retrieved):
          Exact:         0.0537
```

Table 2: Monolingual Arabic for irtba (whole document)

```
      Total number of documents
over all queries
          Retrieved:      2411
          Relevant:       4370
          Rel_ret:         730
      Interpolated Recall -
Precision Averages:
          at 0.00        0.4789
          at 0.10        0.1498
          at 0.20        0.1216
          at 0.30        0.0679
          at 0.40        0.0671
          at 0.50        0.0633
          at 0.60        0.0547
          at 0.70        0.0513
          at 0.80        0.0000
          at 0.90        0.0000
          at 1.00        0.0000
```

```
      Average precision (non-
interpolated) for all rel
docs(averaged over queries)
                        0.0709
      Precision:
        At    5 docs:   0.2059
        At   10 docs:   0.1882
        At   15 docs:   0.1804
        At   20 docs:   0.1691
        At   30 docs:   0.1529
        At  100 docs:   0.0894
        At  200 docs:   0.0671
        At  500 docs:   0.0411
        At 1000 docs:   0.0215
      R-Precision (precision after
R (= num_rel for a query) docs
retrieved):
            Exact:      0.0970
```

As hoped, the title-only run yielded higher early precision, but (also as expected) failed entirely for a number of queries. Of the 50 TREC topics, only 19 yielded any results for this run (indicating that there were no Arabic collection documents with all query terms in the title for the other topics). Topics with some relevant document retrieved included AR37, AR44, AR45, AR48, AR49, AR50, AR51, AR55, AR56, AR61, AR69, and AR74. From this run, we learned that title processing can be effective alone, but fails more often than not if it is the sole basis for retrieval. Combining title retrieval (or differently weighting the title words) with other techniques is indicated.

The base run, using all terms (without differential weighting for title terms), yielded a greater number of relevant documents retrieved (730 vs. 121 for title-only) but lesser early precision and weaker precision over all. Exact precision did not suffer as much, presumably due to a smaller number of failed queries. Nevertheless, only 35 out of 50 topics yielded any results, and 15 of those had no relevant documents. Here, we suffered from working exclusively with the exact match Boolean AND of topic terms. The lack of stemming, plus the lack of any query expansion or partial-match ranking, hurt the set of documents that could be considered and ranked for retrieval.

Overall, these results provide a baseline for VSM-style processing of Arabic documents for mono-lingual runs. Obvious features for inclusion for better results include stemming, query expansion, and differential weighting based on document components such as the title.

Web Track

Two runs for the topic distillation task in the Web track were run. As for the Arabic runs, one was title-only and the other used the entire document. The IRTools indexing took about 4 days for the collection (20GB of HTML documents, about 1.2M documents and 6.37M unique terms). Summary results are in Tables 3 and 4.

Table 3: Web Topic Distillation for irtwt title-only

```
Total number of documents over all
queries
    Retrieved:       901
    Relevant:        737
    Rel_ret:          34
Interpolated Recall - Precision
Averages:
    at 0.00        0.2232
    at 0.10        0.0825
    at 0.20        0.0236
    at 0.30        0.0035
    at 0.40        0.0035
    at 0.50        0.0035
    at 0.60        0.0000
    at 0.70        0.0000
    at 0.80        0.0000
    at 0.90        0.0000
    at 1.00        0.0000
```

```
Average precision (non-
interpolated) for all rel
docs(averaged over queries)
                  0.0237
Precision:
  At    5 docs:   0.0741
  At   10 docs:   0.0519
  At   15 docs:   0.0370
  At   20 docs:   0.0333
  At   30 docs:   0.0284
  At  100 docs:   0.0126
  At  200 docs:   0.0063
  At  500 docs:   0.0025
  At 1000 docs:   0.0013
R-Precision (precision after R (=
num_rel for a query) docs
retrieved):
    Exact:        0.0338
```

Table 4: Web Topic Distillation for irtwb (whole document)

```
Total number of documents over all
queries
    Retrieved:      4728
    Relevant:       1574
    Rel_ret:         141
Interpolated Recall - Precision
Averages:
    at 0.00        0.1153
    at 0.10        0.0740
    at 0.20        0.0465
    at 0.30        0.0243
    at 0.40        0.0235
    at 0.50        0.0174
    at 0.60        0.0042
    at 0.70        0.0010
    at 0.80        0.0010
    at 0.90        0.0000
    at 1.00        0.0000
```

```
Average precision (non-
interpolated) for all rel
docs(averaged over queries)
                  0.0222
Precision:
  At    5 docs:   0.0653
  At   10 docs:   0.0429
  At   15 docs:   0.0408
  At   20 docs:   0.0398
  At   30 docs:   0.0381
  At  100 docs:   0.0286
  At  200 docs:   0.0144
  At  500 docs:   0.0058
  At 1000 docs:   0.0029
R-Precision (precision after R (=
num_rel for a query) docs
retrieved):
    Exact:        0.0372
```

The Web results were not good. Some topics (such as 552) had perfect or near-perfect early precision, while others (such as 551) found no relevant documents at all. Analysis of these results indicates that the main problem is not having heuristics in place to identify good distillation pages, instead relying on regular topic-based matching geared towards term matching. Results were marginally better for the title-only run.

Work to improve results will focus on incorporating document structure into results; in particular the title and heading data which might better indicate good candidates for distillation. In addition, heuristics to look at the document URL itself (which was completely ignored) will help, by flagging shorter URLs are potentially more likely to be good distillation candidates.

Interactive Track

For the interactive track, we are comparing two nearly identical systems using a Google-like text-based interface. Both use the same set of documents, and both make an initial set of candidate documents for ranking using a Boolean AND. They use the Web 02 collection (20GB of HTML from .gov). The difference is that one system uses LSI for ranking results, the other uses VSM with Lnu.Ltc ranking. Document summarization is via Perl modules from CPAN.

Our hypothesis is that the differences in ranking will make no difference in the user experience (i.e., results on measured variables will not be significantly different). We intend this as a base study to explore further variations:

- Systems where the ranked set of documents is different, via automatic query expansion
- Systems where result sets are visualized in a 3D fly-through system

Unfortunately, last year's interactive track was not completed (we intended to compare a text list of results to a browseable category hierarchy), primarily because IRTools was not up to the task. This year, however, the systems are up and running and giving reasonable results. In early 2003, the test interfaces are accessible:

http://underdog.ils.unc.edu/cgi-bin/nph-lsi.cgi (text interface to LSI)

http://underdog.ils.unc.edu/cgi-bin/nph-vsm.cgi (text interface to VSM)

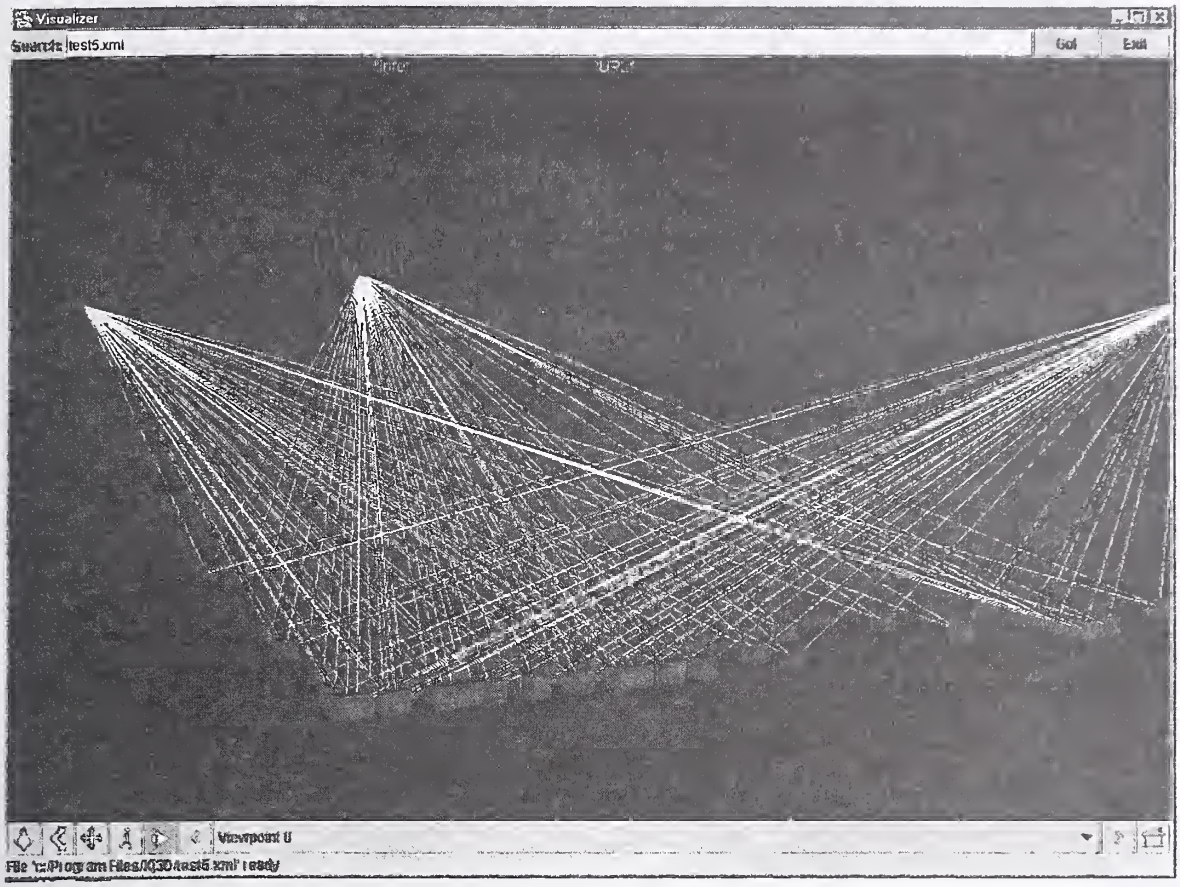
http://underdog.ils.unc.edu/cgi-bin/nph-query.cgi (VSM with database select)

The 3D interface is implemented in Web3D (essentially, Web3D is a modern VRML '97 implemented over Java3D). This interface runs by accepting user queries, running them against the LSI module of IRTools, then displaying the resulting set of term and document locations and relationships. A simple XML structure is used to communicate between the visualizer and the server.

Note that the LSI applied is only to the Boolean AND of search terms, or a slightly expanded set of search terms. This is done while the user waits (usually within a few seconds, depending on the number of terms and documents being considered). For larger-scale LSI, we have constructed some very large LSI spaces into which queries may be mapped (such spaces are also good for query term expansion). For general visualization of search set results using only documents that contain the query terms, the technique described here seems to work well.

We will evaluate this visual interface in several contexts, and determine whether it is effective in determining relations among documents in post-search result sets.

Figure 1: 3-word query with lines denoting set membership. Clickable documents appear as small cubes.



File 'c:/Program Files/IQ3D/test5.xml' ready

GridIR

Grid computing is an important advance in computational techniques. It has some concepts in common with distributed computing and with massively parallel computing, but many added features. GridIR is IR on the computing grid. The author and his colleagues have worked to form a GridIR working group under the auspices of the Global Grid Forum (http://gridforum.org). We believe that GridIR offers important advantages to IR researchers, and will make experimental and mainstream IR systems more usable and better suited for large-scale research.

Grid computing has a security model built in, making GridIR suitable for publishing partial extranets or implementing security at the query, collection, document or user level. We are currently working on a draft requirements document for the GGF for delivery in spring 2003, and welcome input and efforts from other IR researchers. Reference systems for GridIR will include IRTools and Amberfish, and we welcome others. Our goal is to develop a set of actual standards for GridIR (under the GGF, following a rulemaking procedure similar to the IETF). We are building on knowledge from Z39.50 and other efforts, and hope to enable a far higher level of interoperability among content maintainers, searchers and IR systems than is now available.

Visit the GridIR Web site to learn more: http://www.gridir.org.

Conclusion

IRTools continues to develop, and despite results being late was able to handle the Web and Arabic tracks with relative ease. Continued work will make IRTools more usable, and integration with the GridIR reference implementation will help to shake out bugs and shape future developments. CLIR continues to be a focus, with new modules for Chinese and Arabic recently added.

IR researchers are urged to consider GridIR as a possible activity. Credibility and buy-in from IR systems developers, vendors, scholars, etc. will help make GridIR as beneficial as possible.

Acknowledgements

This work was supported in part by the National Science Foundation (NSF award #CCR-0082655).

Many students and employees have worked on the software for this year's results. Their efforts are significant and valued. Some of these fine people include: Li Chen, Nassib Nassar, Mao Ni, Li Wen, and Yuehong Wang. Alan Hudson and Yumetech, Inc. of Seattle have performed work on the visualizer, under a grant from the E.S.P. Das Educational Foundation.

Report on the TREC-11 Experiment: Arabic, Named Page and Topic Distillation Searches

Jacques Savoy, Yves Rasolofo

Institut interfacultaire d'informatique, Université de Neuchâtel (Switzerland)
E-mail: {Jacques.Savoy, Yves.Rasolofo}@unine.ch

Summary

This year we took part in the Arabic cross-language information retrieval track (for us limited to monolingual Arabic retrieval) and also in both named page and topic distillation searches. In the last two tasks, we made use of link anchor information and document content in order to construct Web page representatives. This document representation uses multi-vectors in order to highlight the importance of both link anchor information and document content.

Introduction

Today the IR community is faced with a new paradigm and many exciting challenges with regards to Web page searches. Some of these include: managing huge volumes of documents via distributed IR models, crawling across the Web in order to find appropriate Web sites to index, accessing documents written in various languages, measuring the quality or authority of available information, providing answers to user requests that are often very short and expressed in ambiguous terms, satisfying a large range of search types (ad hoc, question-answering, location of online services, topic distillation, known-item and interactive searches for specific document types, or satisfying specific geographical or time constraints).

In this context, the first part of this paper presents our monolingual Arabic retrieval model. Section 2 describes our procedures for indexing and retrieving Web pages based on two document representations, and our distributed indexing framework based on the Okapi probabilistic model. Section 3 explains the IR approach we use when combining both Web page content and anchor information when searching for specific named pages. Finally, Section 4 describes how our IR scheme can be used within the context of topic distillation task.

In order to evaluate our hypothesis, we used the SMART system as a testbed, implementing various vector-space IR schemes and the Okapi probabilistic model (Robertson et al., 2000). This year our experiments were conducted on an Intel Pentium III/600 (memory: 1 GB, swap: 2 GB, disk: 6 x 35 GB) and all experiments were fully automated.

1. Arabic Information Retrieval

During the last two CLEF evaluation campaigns we suggested various IR models and tools for handling different European languages (Savoy, 2002a; 2002b). This year we expanded upon our knowledge by adding the Semitic language family, which includes Arabic.

The IR model we are proposing for Arabic text searches involves indexing both documents and queries, based on the words described in Section 1.1, or using n-gram segmentation, as presented in Section 1.2. Section 1.3 shows how we can combine result lists provided using various indexing and searching schemes that process the same document collection. The last section provides an account of the retrieval effectiveness achieved by various IR models and also that of various combined approaches. The diverse IR tools may be seen on our Web site (www.unine.ch/info/clef/).

1.1. Word-Based Indexing

In order to effectively search Arabic documents, we first convert and normalize the Arabic Unicode characters into Latin letters (a technique used in Malta where the Arabic language is written using the Latin alphabet). Due to variations in morphological rules or geographical traditions however many Arabic characters have more than one Unicode representation. For example there are different forms of alef (alef madda "آ", alef hamza above "أ", alef hamza below "إ"), transliterated within our approach into the same letter "A" (see Table 1). Our conversion procedure is based on but is not identical to that used in previous work by Darwish et al. (2002). There are of course some questionable assignments such as the hamza letter "ء", which is considered equivalent to the alef maksura "ى". In this phase some Unicode characters have also been removed (e.g., diacritic marks that are usually optional in newspapers such as tatweel "ـ", fathatan "ً" or various punctuation marks "؟". These diacritic marks may however be important in other contexts in order to resolve underlying ambiguity (e.g., in legal documentation)).

In a second step we ignore words appearing in our Arabic stopword and Arabic stoplist (the latter contains 347 words, available at www.unine.ch/info/clef/).

In a third step we automatically remove both prefixes and suffixes to form our Arabic stems. These relatively light stemming approaches are similar to those suggested by Larkey & Connell (2002) or Lackey et al. (2002). As shown in Table 2, our stemmer "stem2" is more conservative while our "stem3" represents a more aggressive affix-stripping process. The word length must be greater than a given threshold (between 4 and 6 letters) before we will remove a given affix. Some of our rules are questionable and our stemmers must be viewed, as they are only a first draft.

Arabic	Latin	Arabic	Latin
ء ى	Y	إ أ آ ا	A alif
ؤ و	w waw	ئ	y yeh
ب	b beh	ة	p teh
ت	t teh	ث	v theh
ج	j jeem	ح	H hah
خ	x khah	د	d dal
ذ	O thal	ر	r reh
س	s seen	ش	P sheen
ص	S sad	ض	D dad
ط	T tah	ظ	Z zah
ع	E ain	غ	g ghain
ف	f feh	ق	q qaf
ك	k kaf	ل	l lam
م	m meem	ن	n noon
	h heh	ي	X yeh
ز	z zain		

Table 1. Our Arabic letter transliterations

	Remove from front	Remove from rear
stem2	fAl, XAl, bAl, wAl, Al, w, Y	An, At, hA, Yp, Yh, Yn, wn, Y, p, h
stem3	wAl, fAl, bAl, Al, ll, bt, yt, lt, mt, tt, wt, st, nt bm, lm, wm, km, fm, wA, fA, lA, bA, wy, ly, sy, fy, t, y, m, b, n, l, k, w, A, f	km, tm, At, An, wn, wh, hn, hm, wA, tA, hA, nA, tk, ty, th yn, yh, yp p, h, y, t, k, A

Table 2. Main rules used by our two Arabic stemmers

1.2. N-gram Indexing

As an alternative procedure we can index Arabic documents using 3-gram (or tri-gram) indexing (Darwish & Oard, 2002). In this case, each word is replaced by a set of three-letter sequences. For example the word "document" will be replaced by {"doc", "ocu", "cum", "ume", "men" and "ent"}. In our current implementation we do not stem words before splitting them into tri-grams and we also remove very frequent tri-grams (obtained from our stopword list).

1.3. Data Fusion

We use a single search model (or engine) when searching document collections. We might however suggest sending the request to different search engines that handle the same document collection but that use different indexing or scarch schemes. Finally, once we have obtained result lists from these various search engines, we need to merge them in an effective manner (data fusion problem). Thus, even though certain degrees of rctrieval effectiveness may be attributed to each search approach, combining the result lists might provide better average precision. If we were to use RSV_k to denote the retrieval status value (or document score) for a given document retrieved by the *kth* search engine, Fox & Shaw (1994) suggest using various operators (see Table 3) and show that the best performance can be achieved using "combSUM".

combMAX	MAX (RSV_k)
combMIN	MIN (RSV_k)
combSUM	SUM (RSV_k)
combANZ	SUM (RSV_k) / # of nonzero (RSV_k)
combNBZ	SUM (RSV_k) * (# of nonzero (RSV_k))
combRSV%	SUM (RSV_k / maxRSV)
combRSVn	SUM[(RSV_k-minRSV)/(maxRSV-minRSV)]

Table 3. Data fusion strategies

Of course we might also employ the round-robin merging strategy whereby we take the first retrieved item from the first result list, then the first retricved document from the second list, etc., and finally the first item from the last result list and then back again to the first results list, thus providing the next item to be put in the final list. Duplicates encountered in this process are simply ignored.

1.4. Evaluation

We evaluated various vector-space schemes (see Appendix 1 for detailed specifications of these models) together with the Okapi probabilistic model. As shown in Table 4a, we also evaluated our two light stemmers and the tri-gram indexing scheme using short queries ("Title" or T), medium-size queries (constructed using the Title and Descriptive logical sections or TD) or long queries (based on Title, Descriptive and Narrative sections or TDN).

An examination of this data shows that the best average precision is obtained using the Okapi model while second best results are usually obtained using the vector-space model "Lnu-ltc", and the "dtu-dtn" scheme usually ranks third. Moreover, it seems that our "stem2" stemmer performs slightly better than the more aggressive "stem3" procedure (the mean difference over 10 retrieval models was 4.7% for T queries, 6.1% for TD queries, and 7.4% for TDN queries). On average, the tri-gram indexing scheme seems to be a little bit less effective than word-based indexing (using the "stem3" or "stem2" stemming approaches). Note however that with the Okapi model, the tri-gram approach performed better for shorter queries (Title only) or TD requests.

From previous evaluations on different European languages (Savoy, 2002a), it is clearly apparent that requests containing more search terms provide improved average precision (from "Title" to TD, with a mean improvement of around 13.3% and a mean difference between "Title" and TDN of around 17.5%). With the Arabic corpus these differences appear to fall within a similar range, as shown in Table 4a. For example, using the Title only evaluation as a baseline, performance can be improved by about 9.9% for TD queries (mean over 10 retrieval models, using stem2) or 4.9% with TD requests (mean over 10 retrieval models, using tri-grams). When comparing short request query performances (Title only and TDN), the mean difference over 10 IR models is around 18.6% (stem2) or 15.6% (stem3). For tri-gram models however the average precision differences are around -2.1%, due to the poor performance by the "nnn-nnn" and "bnn-bnn" approaches during TDN queries.

	Average Precision								
	Title			TD			TDN		
Model	Word stem2	Word stem3	3-grams no stem	Word stem2	Word stem3	3-grams no stem	Word stem2	Word stem3	3-grams no stem
Okapi-npn	**27.41**	**26.09**	**28.77**	**30.51**	**29.22**	**31.45**	**32.87**	30.40	**29.95**
Lnu-ltc	25.80	24.93	25.95	29.88	28.68	29.79	32.33	**30.41**	29.49
dtu-dtn	24.92	24.35	23.98	27.25	25.22	27.96	28.45	27.30	26.30
atn-ntc	22.71	21.30	22.76	24.44	22.42	24.00	25.98	25.47	22.44
ltn-ntc	24.19	22.94	22.65	26.45	24.77	24.50	27.94	26.60	21.58
lnc-ltc	20.66	19.55	20.46	24.77	23.38	25.10	29.58	27.05	27.39
ntc-ntc	20.27	19.09	19.64	23.03	21.46	23.92	25.38	23.39	23.26
ltc-ltc	18.41	17.90	19.73	21.33	20.30	23.95	26.25	24.43	25.40
nnn-nnn	12.65	12.11	8.22	13.84	13.21	6.31	14.84	13.60	5.10
bnn-bnn	12.47	11.64	11.29	10.86	9.92	5.90	8.54	7.00	1.62

Table 4a. Average precision of various IR models using the Arabic corpus (monolingual)

	Average Precision								
	Title			TD			TDN		
#doc/#term	Word stem2	Word stem3	3-grams no stem	Word stem2	Word stem3	3-grams no stem	Word stem2	Word stem3	3-grams no stem
Okapi-npn	27.41	26.09	28.77	30.51	29.22	31.45	32.87	30.40	29.95
d=5 / t=10	31.51	31.13	32.30	33.62	32.46	33.56	35.21	33.06	32.66
d=5 / t=20	32.45	31.91	32.40	34.15	33.31	34.60	35.75	33.59	33.67
d=5 / t=30	32.98	31.85	32.49	34.23	33.24	35.30	36.25	33.86	33.96
d=5 / t=40	33.34	31.75	33.37	34.47	33.27	35.68	36.36	34.07	34.19
d=5 / t=50	33.26	31.64	33.35	34.45	33.20	35.78	36.47	34.12	34.07
d=5 /t=100	33.32	31.54	32.62	34.64	33.19	36.19	36.40	**34.27**	**34.34**
d=5 /t=150	33.07	31.32	32.23	34.48	33.22	**36.24**	36.39	33.87	34.17
d=10 /t=10	32.39	31.57	32.75	34.03	32.81	33.52	35.27	32.68	32.37
d=10 /t=20	33.35	32.41	34.44	34.78	33.53	34.72	36.01	33.69	32.69
d=10 /t=30	33.82	32.78	**34.78**	35.23	33.88	35.04	36.39	33.60	32.85
d=10 /t=40	34.13	**32.97**	34.71	35.52	**34.08**	35.39	36.46	33.49	33.03
d=10 /t=50	**34.20**	32.89	34.54	35.66	34.08	35.50	36.42	33.40	33.14
d=10/t=100	34.00	32.17	34.56	**35.85**	33.69	35.70	36.52	33.56	33.42
d=10/t=150	33.75	31.76	34.35	35.46	33.40	35.36	**36.55**	33.41	33.41
d=25 /t=10	32.06	31.33	31.91	33.75	32.21	32.72	34.99	32.53	31.61
d=25 /t=25	33.09	32.81	32.02	34.59	33.03	33.66	35.74	33.04	32.07
d=25 /t=50	33.78	32.96	32.55	35.24	33.21	34.01	36.01	33.54	32.47
d=25/t=100	33.88	32.90	32.57	35.42	33.09	34.11	36.07	33.45	32.52

Table 4b. Average precision using blind query expansion (Okapi model, Arabic corpus, monolingual)

	Average Precision								
Model	Title no expand	TD no expand	TDN no expand	Title + expand 1 / 1 / 1	TD + expand 1 / 1 / 1	TDN + expand 1 / 1 / 1	Title + expand 1 / 1 / 1.5	TD + expand 1 / 1 / 1.5	TDN + expand 1 / 1 / 1.5
Best single	28.77	31.45	32.87	34.78	36.24	36.55	34.78	36.24	36.55
Round-robin	28.68	31.40	32.28	35.21	36.81	36.59	35.21	36.81	36.59
combRSVn	30.19	33.13	**33.60**	**36.54**	**36.90**	36.94	**36.75**	**37.16**	**36.98**
combMAX	**30.79**	**33.58**	31.78	35.25	36.56	**37.12**	34.44	36.57	34.85
combSUM	29.69	33.03	33.23	35.98	36.65	36.63	36.31	37.01	36.59
combRSV%	29.02	32.63	32.85	36.00	36.33	36.27	36.04	36.61	36.09
combNBZ	29.07	32.78	33.02	35.94	36.58	36.36	36.08	37.05	36.48
combANZ	27.26	28.34	27.67	33.83	32.78	33.67	27.17	22.54	23.88
combMIN	20.52	20.77	20.09	30.29	27.14	27.85	18.41	11.94	14.26

Table 5. Average precision of various data merging strategies (Arabic corpus, monolingual)

Run name	Query	Okapi stem3	Okapi stem2	Okapi 3-grams	Average precision
UniNE1	T-D	doc=10 / term=15	doc=10 / term=75	doc=10 / term=20	**37.12**
UniNE2	T	doc=10 / term=40	doc=5 / term=20	doc=25 / term=15	35.72
UniNE3	T-D-N	doc=5 / term=50	doc=10 / term=40	doc=10 / term=20	38.07
UniNE4	T-D	doc=10 / term=15	doc=10 / term=75	doc=10 / term=20	36.60

Table 6. Specifications and evaluation of our official monolingual Arabic runs

Pseudo-relevance feedback (or blind-query expansion) has proven to be a useful technique for enhancing retrieval effectiveness. In this study, we adopted Rocchio's approach (Buckley et al., 1996) with $\alpha = 0.75$, $\beta = 0.75$ whereby the system was allowed to add t terms extracted from the k best ranked documents obtained from the original query. We used the Okapi probabilistic model to evaluate this proposal and then enlarged the query by 10 to 75 terms taken from the 5 or 10 best-ranked articles (see Table 4b). From examining the data in this table we were able to conclude that overall blind-query expansion does indeed improve retrieval performance. For example, with short requests the improvement is +21.5% when applying the stem2 stemmer, +23.1% with stem3 and +16% with the tri-gram model.

Finally, we evaluated various data fusion strategies that might be employed to improve retrieval effectiveness. In our case we used the same document collection and simply submitted the same request to our three search engines (stem2, stem3, and tri-grams). Based on the data in Table 5, it appears that data fusion based on combRSVn performs better when blind query expansion is taken into account. The combMAX strategy seems to be the more appropriate solution when we ignore pseudo-relevance feedback. However, both combRSVn and combSUM seem to be more robust data fusion operators. We should however mention that the percentage improvement over the best single approach is not really significant (e.g., in Title only queries without query expansion, the combRSVn increases the average precision by +4.9% compared to +2.2% when using TDN requests). Finally, the last three columns of Table 5 show the results of the same three individual runs where we instead multiplied each document score obtained from the tri-gram model by 1.5, without modifying document scores for both the word-based indexing schemes.

Table 6 lists the exact specifications for our official runs. These runs were carried out using different numbers of documents and terms during blind query expansions but all runs were built using the combRSVn operator and multiplying the tri-gram document scores by 1.5.

2. Our Okapi Search Model

As shown in the preceding section, the Okapi search model provides significantly greater retrieval effectiveness. However, in order to manage the Web collection (1,247,753 documents as extracted from the .GOV domain, or about 18.1 GB of data), we needed to modify this search model for two reasons. First, we wanted to incorporate two document representatives for each Web page, and secondly we needed to distribute the inverted file in order to respect the 2 GB limit.

When using multiple document representations, the retrieval status value (or document score denoted RSV(D_i)) was calculated as follows (inner product):

$$\text{RSV}(D_i) = \sum_{j=1}^{m} w_{ij} \cdot qw_j$$

within which w_{ij} indicates the weight assigned to the term T_j in the document D_i, qw_j the indexing weight assigned to the same term in the current query and m the number of search keywords.

When processing two (or more) document representations, we estimated the degree of similarity between the document D_i and the current query as a linear combination of the inner product of the two document representations to be given as:

$$RSV(D_i) = \alpha \cdot \sum_{j=1}^{m} w_{ij}^{(1)} \cdot qw_j + (1-\alpha) \cdot \sum_{j=1}^{m} w_{ij}^{(2)} \cdot qw_j \quad (1)$$

where $w^{(1)}_{ij}$ indicates the weight attached to the term T_j in the document D_i in the first document representation (and $w^{(2)}_{ij}$ for the second document surrogate), and α a parameter used to assign a comparative importance to the first document representative as relative to the second.

Thus, by assigning α a value close to 1.0, we give more importance to the first document representation. At the limit, setting $\alpha = 1.0$ implies that we ignore the second document surrogate.

Creating a single inverted file from a collection of around 18 GB might be impossible using a 32-bit system (e.g., Linux). To overcome this limit, we will concentrate on the last scheme, and in this case we will follow the approach described in (Rasolofo & Savoy, 2003), whereby we merge the result lists obtained from searching different collections (collection fusion problem). This is achieved by using the document scores computed by each collection as a key for the merging and sorting process.

3. Named Page Searching

When submitting a request to a search engine, sometimes users do not want a ranked list of Web pages regarding a particular topic but rather they would prefer the location of an underlying service or known-item (usually presented in a short list of the most probable locations). For example, an appropriate answer to the requests "US passport renewal", "Maryland unemployment insurance benefits" or "FBI's most wanted list" does not consist of a ranked list of documents about these subjects but rather the site(s) containing the required form/information/list. To accomplish this goal we need to implement an IR system that can retrieve a short number of pages (at the limit, only one) corresponding to the user's request. In this context of known-item search, the underlying IR system must clearly place the emphasis on precision.

3.1. Search Models

As a basis for our search model we used the Okapi model as described in Section 2. Our first document representative was based on information found in the Web page and its corresponding <TITLE> and <META> tags ("keywords" and "description"). Web pages might of course also contain links and their anchor texts (or anchor texts for outgoing links) and this combined set of internal textual information would thus form the first representatives of these pages.

On the other hand, previous studies (Craswell et al., 2001), (Westerveld et al., 2002), (Kraaij et al., 2002) have shown that anchor texts from other Web pages pointing to the current page provide compact and often accurate descriptions of the current page's content. Thus link anchor texts extracted from all Web pages pointing to the current page were concatenated to form our second document representative. To this second surrogate we also added the text contained in the current page's <TITLE> tag (with the text delimited by this tag appearing in both document representatives). Finally, we might also consider the URL content (or more precisely, the similarity between the URL text and the request, or also the URL length). This additional source of information has not taken into account in our current search models.

3.2. Evaluation

In this IR search model based on two document representatives, we first needed to determine the relative importance assigned to the first document representative (based on internal Web page content) as compared to the weight attached to the second document surrogate (based mainly on link anchor texts from Web pages pointing to the current one). This relative importance for each surrogate is controlled by the α parameter (see Section 2). When $\alpha = 1.0$, we would only account for internal textual representation while when setting $\alpha = 0.5$, we would attribute equal importance to both document representatives.

Number of queries	150
Number of relevant doc.	170
Mean rel. doc. / request	1.133
Standard deviation	0.378
Median	1
Maximum	3 (q#: 9, q#: 145)
Minimum	1

Table 7. Relevance judgment statistics (named page searches, TREC-11)

Our evaluation will be based on the mean reciprocal rank (MRR) of the first correct answer found by the system. Table 7 depicts statistics on the relevance assessments of this test-collection, clearly showing that we usually obtain one correct answer per topic. For each of the 150 queries we considered only the first 100 retrieved items. As seen in Table 8, the best α value seems to be around 0.6, thus assigning a little more weight to internal representation.

Run name	MRR	# in top 10	# not found
α = 0.0	0.5046	99 (66.0%)	27 (18.0%)
α = 0.1	0.5507	106 (70.67%)	24 (18.0%)
α = 0.2	0.5929	109 (72.67%)	22 (14.67%)
α = 0.3	0.6366	115 (76.67%)	16 (10.67%)
α = 0.4	0.6491	120 (80.0%)	12 (8.0%)
α = 0.5	0.6688	120 (80.0%)	7 (4.67%)
α = 0.6	**0.6800**	122 (81.33%)	**7** (4.67%)
α = 0.7	0.6775	**125** (83.33%)	7 (4.67%)
α = 0.8	0.6772	124 (82.67%)	9 (6.0%)
α = 0.9	0.6511	121 (80.67%)	11 (7.33%)
α = 1.0	0.5867	116 (77.33%)	16 (10.67%)

Table 8. IR model evaluation for various combinations of two document representatives (no stemming)

Run name	MRR	# in top 10	# not found
α = 0.0	0.4991	99 (66.0%)	25 (16.67%)
α = 0.3	0.6163	118 (78.67%)	15 (10.0%)
α = 0.5	0.6506	120 (80.0%)	9 (6.0%)
α = 0.7	0.6735	123 (82.0%)	6 (4.0%)
α = 0.8	0.6710	122 (81.33%)	6 (4.0%)
α = 1.0	0.5771	116 (77.33%)	13 (8.67%)

Table 9. IR model evaluation for various combinations of our two document representatives (S-stemming)

When high precision results are required for indexing documents or requests, it is usually not a good idea to include a stemming procedure. However a very light stemming procedure might only be adopted when removing the final "s" in English (such stemming is called "S-stemming" (Harman, 1991)). Thus the words "house" and "houses" will be reduced to the same root while the term "housing" will be treated as a different indexing unit.

A comparison of results depicted in Table 9 (S-stemming) to those in Table 8 (no stemming) indicates that performance differences are rather small. Better performances can however always only achieved from IR approaches that ignore the stemming phase.

Finally, Table 10 provides a summary description of our four official runs. Only the UniNEnp3 run needed an additional comment. This run was based on UniNEnp1 and after obtaining a ranked list, we reranked the first ten retrieved items according to the number of matches between the query terms and the corresponding Web page's title field (however, such a strategy does not improve retrieval effectiveness).

Run name	MRR	Description
UniNEnp1	**0.636**	No stemming, α = 0.3
UniNEnp2	0.616	S-stemming, α = 0.3
UniNEnp3	0.625	Reranking the first 10 items
UniNEnp4	0.504	No stemming, α = 0.0

Table 10. Description of official named-page runs

4. Topic Distillation Searches

Under the label "topic distillation", we had to implement an IR scheme able to find a list of key resources on a given topic. Explicitly defining what does or does not constitute a good key resource is however difficult, and each definition seems to become more ambiguous. Of course, Web pages with appropriate content might be considered as good key resources and we could retrieve them using a classical IR model. On the other hand, key resources may also be good hubs (or Web pages pointing to different pages containing pertinent content with respect to the submitted request). Moreover, if a Web page is linked to two, three or more sons having a high degree of similarity with the request, it seems more appropriate to return this father page rather than the two, three of more sons. More generally however returning many pages extracted from the same Web site would not be viewed as a wise strategy. Thus to suggest a proper solution for this specific task, we decided to employ different strategies capable of pointing to reliable starting points for browsing rather than simply retrieving Web pages with good content. An overview of such strategies that might be applied in a Web environment can be found in Savoy & Rasolofo (2001).

4.1. Search Models

As for the task of named page searching, we built two document representatives for each Web page contained in the .GOV collection. The first representative accounted for Web page content along with its <TITLE> and <META> tags ("keywords" and "description") plus all link anchor texts extracted from other pages pointing to this current page. The second document representative was built from the text delimited by the <TITLE> tag together with link anchor texts from all outgoing links. These two document representations may be useful both for accounting for the content of both the Web page (first surrogate) and other pages accessible within a one-click distance from the current page (our second representative).

Once the pages are retrieved, we followed hyperlinks coming into them in order to define proper starting points for browsing (in this case we followed existing hyperlinks in the reverse orientation). To retrieve these starting points we used our spreading activation (SA) searching scheme (Savoy, 1996), (Crestani & Lee, 2000), (Savoy & Picard, 2001). Using this method, document scores initially computed by the IR system (denoted RSV(D_i)), are propagated to the linked documents through a certain number of cycles, using a propagation factor. We used a simplified version with only one cycle and a fixed propagation factor λ for all links. Thus the final retrieval status value for a document D_i linked to k documents is computed using the following equation:

$$RSV'(D_i) = RSV(D_i) + \lambda \cdot \sum_{j=1}^{k} RSV(D_j) \qquad (3)$$

When trying in our experiments to extract the proper starting sites for browsing, we only considered all incoming links for each of the k best-ranked documents (in this paper the constant k was fixed to 200 and the parameter λ to 0.35).

As an alternative, we assumed that the first k top-ranked items would form a "root set" or a kemel of pertinent pages from which we could consider all incoming and all outgoing links in order to form an extended set (called the base set) of pages that might be of interest for a given topic. Based on Kleinberg's HITS algorithm, we assumed that a Web page pointing to many other information sources must be viewed as a "good" hub while a document with many Web pages pointing to it must be viewed as a "good" authority. Likewise, a document that points to many "good" authorities is an even better hub while a Web page pointed to by many "good" hubs is an even better authority (Kleinberg, 1998).

For document D_i after c+1 iterations, the updated formulas for the hub and authority scores $H^{c+1}(D_i)$ and $A^{c+1}(D_i)$ are:

$$A^{c+1}(D_i) = \sum_{D_j = parent(D_i)} H^{c}(D_j)$$

$$H^{c+1}(D_i) = \sum_{D_j = child(D_i)} A^{c}(D_j)$$

which is computed for the k best-ranked documents (defincd as the root set) retrieved by a classical search model, together with their children and parents (which defined thc base set). The hub and authority scores were updated for five iterations (while the ranking did not change after this point), and a normalization procedure (dividing each score by the sum of all square values) was applied after each step.

As other possibilities, we might consider the PageRank algorithm (Brin & Page, 1998) or probabilistic argumentation systems (Picard, 1998).

4.2. Evaluation

In order to evaluate the performance of a topic distil!ation IR schcme, we could use the precision achieved after retrieving 5 or 10 documents (under the labels "Prec@5" or "Prec@10") together with the number of relevant items retrieved (out of a total of 1,574 for the 49 queries included in the .GOV collection).

Number of queries	49
Number of relevant doc.	1,574
Mean rel. doc. / request	32.122
Standard deviation	37.33
Median	22
Maximum	188 (q#: 558)
Minimum	1 (q#: 588)
Number of distinct roots / query	
Mean	9.429
Standard deviation	15.27
Median	13
Maximum	64 (q#: 596)
Minimum	1 (q#: 581)
URL length 1	31
length 2	194
lcngth 3	536
length 4	402
length 5	263
length 6	110
length 7 and more	38
# pertinent items file	1,380
# pertinent items path	194

Table 11. Relevance judgment statistics (topic distillation searching task, TREC-11)

Table 11 shows various statistics based on relevance assessments. The mean number of relevant items (or key resources) per request is 32.122. From considering the number of distinct roots (e.g., the first part of an URL, e.g., "trec.nist.gov"), we find that in mean, there were 9.4 different roots per query (for Query# 581, all relevant items coming from the root page "www.cancer.gov"). On the other hand, for Query# 558, we found 26 relevant pages extracted from the root page "www.whitehouse.gov" (and of these, 25 were from "www.whitehouse.gov/news/releases/2001/").

In our first set of experiments, we evaluated our extended Okapi IR model (see Section 2). By varying the value attached to the α parameter, we assigned more or less weight to each document representation. More precisely, when we set α = 0.0, we accounted for text delimited by the <TITLE> tag and all link anchor texts from outgoing links. In other words, we viewed the page as a good starting point for browsing (limited howevcr to one-click distance). On the other hand, when α = 1.0, our search model was based on Web page content and from the various link anchor texts contained in all pages pointing to this particular document.

Table 12a displays the various results produced by our IR model (without stemming) when varying the relative importance of each document representative. From this data, the best α value seemed to be around 0.9, based upon the precision achieved after 10 retrieved items (or 0.7 for 5 retrieved records). Thus, our first representation (content-oriented) seems to be more valuable for this specific IR task. Data in Table 11 seems to confirm these findings, given the various statistics on relevance assessments used in this task. For example, of the 1,574 pertinent items, 1,380 (or 87.7%) correspond to a filename while only 194 (or 12.3%) to subdirectories or path entries (URLs ending with a "/" or with "index.htm" or similar terms). Moreover, URLs of unitary length (or roots) correspond to only 31 (or 2%) relevant items.

Run name	Prec@5	Prec@10	rel. & retr.
α = 0.0	15.92	13.06	457
α = 0.1	17.14	14.69	562
α = 0.2	18.37	15.92	626
α = 0.3	18.78	17.35	683
α = 0.4	20.82	17.14	793
α = 0.5	21.22	17.96	926
α = 0.6	22.04	19.39	982
α = 0.7	**24.08**	19.59	973
α = 0.8	22.86	21.43	**991**
α = 0.9	22.86	**21.63**	965
α = 1.0	23.67	18.37	919

Table 12a. Evaluation of various document representatives combinations (no stemming. TD queries)

Run name	Prec@5	Prec@10	rel. & retr.
α = 0.0	11.84	9.39	635
α = 0.2	18.37	13.47	877
α = 0.4	21.63	16.12	1,217
α = 0.6	20.82	18.16	**1,231**
α = 0.8	**22.04**	**18.98**	1,159
α = 1.0	22.04	17.35	1.064

Table 12b. Evaluation of various document representatives combinations (no stemming, TDN queries)

When we considered longer queries (built using the Title, Descriptive and Narrative logical sections), retrieval performance seemed to decrease relative to the precision achieved upon retrieving 5 or 10 items. Of course this value clearly increases for longer requests, as shown by the number of relevant and retrieved records (last column of Table 12b).

Our UniNEdi1 run is based on short requests (Title only) while our UniNEdi3 run is based on the same processing but for TDN queries. For both runs, after retrieving content-based Web pages using our extended Okapi model, we applied spreading activation with λ = 0.35 for the first k = 200 top-ranked items. Following this stage, we pruned the retrieved URL (keeping only three URLs per site).

Using the SA method and based on the best run data shown in Table 12a, we tried various parameter settings as depicted in Table 13. Clearly, the propagation factor λ must be smaller than 0.35, and the SA must be limited to the first 50 best-ranked items (instead of k = 200).

Parameters	Prec@5	Prec@10	rel. & retr.
no stem, α = 0.9	22.86	21.63	965
λ = 0.01, k = 50	23.27	21.43	1,020
λ = 0.025, k = 50	**25.71**	**21.84**	**1,020**
λ = 0.05, k = 50	25.31	21.63	1,020
λ = 0.1, k = 50	22.86	19.39	1,020
λ = 0.15, k = 50	20.82	18.37	1,020
λ = 0.2, k = 50	19.59	16.73	1,020
λ = 0.1, k = 25	22.86	19.39	1,000
λ = 0.1, k = 75	22.04	18.37	1,029
λ = 0.1, k = 100	20.00	18.16	1,036
λ = 0.1, k = 200	15.10	15.71	1.051

Table 13. Evaluation of various parameter settings for the spreading activation approach

For the UniNEdi2 run, we applied the Kleinberg's HITS algorithm in order to define hub and authority pages (k = 200), and to form our ranked list we summed the hub and authority scores of each Web page, defining the new document score. Finally we pruned the retrieved URL.

Using the best run from Table 12a as the starting point, we varied the number k of the top-ranked items included in the root set from the HITS method, as shown in Table 14. The data in this table seems to clearly indicate that in this task the HITS algorithm does not perform well. whatever the value of k, whether we account for the hub score, the authority score or both.

Finally, Table 15 provides a summary description of our five official runs, all of which were created without a stemming procedure. Searching for good browsing starting points when using the SA or Kleinberg approaches clearly fails. or more precisely searching key resource does not means searching for browsing proper starting points.

Parameters	Prec@5	Prec@10	rel. & retr.
no stem, α = 0.9	22.86	21.63	965
k = 50, hub score	3.67	3.27	526
k = 50, auth. score	6.12	4.90	526
k = 50, both	2.86	3.27	526
k = 100, hub score	2.86	2.45	684
k = 100, auth. score	**5.31**	**3.88**	679
k = 100, both	2.45	2.45	683
k = 150, hub score	2.04	1.84	762
k = 150, auth. score	4.90	3.88	742
k = 150, both	2.04	2.04	765
k = 200, hub score	1.22	1.22	771
k = 200, auth. score	4.49	2.86	717
k = 200, both	1.22	1.22	730
k = 300, hub score	0.41	0.82	643
k = 300, auth. score	3.67	2.45	576
k = 300, both	0.82	0.83	598

Table 14. Evaluation of different parameter settings for the HITS algorithm

Run name	Prec@10	description
UniNEdi1	8.37	UniNEdi5 + SA (λ=0.35)
UniNEdi2	3.27	UniNEdi5 + HITS
UniNEdi3	7.76	TDN, no stem, α = 0.7, SA
UniNEdi4	14.29	UniNEdi5 + reranking
UniNEdi5	19.59	no stemming, α = 0.7

Table 15. Description of our official named page runs

Only the UniNEdi4 run needs any additional comments. This run is based on UniNEdi5 and after we obtained a ranked list, we computed and sorted the Web sites according to number of pages present in the top 50 best-ranked items. Following this step, we selected pages from those sites having the greatest number of matches between the query terms and the underlying URL texts (however, this selection and reranking procedure did not improve the retrieval effectiveness).

Acknowledgments

The authors would like to thank S. Abdou for his help in understanding some aspects of the Arabic language and C. Buckley from SabIR for allowing us the opportunity to use the SMART system. This rescarch was supported by the SNSF (Swiss National Science Foundation) under grants 21-58'813.99 and 21-66'742.01.

References

Brin, S. & Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. Proceedings WWW8, 107-117.

Buckley, C., Singhal, A., Mitra, M. & Salton, G. (1996). New retrieval approaches using SMART. Proceedings TREC-4, NIST Publication #500-236, 25-48.

Craswell, N., Hawking, D. & Robertson, S. (2001). Effective site finding using link anchor information. Proceedings ACM-SIGIR'2001, 250-257.

Crestani, F. & Lee, P.L. (2000). Searching the Web by constrained spreading activation. *Information Processing & Management*, 36(4), 585-605.

Darwish, K. & Oard, D.W. (2002). Term selection for searching printed Arabic. Proceedings ACM-SIGIR'2002, 261-268.

Darwish, K., Doermann, D., Jones, R., Oard, D.W. & Rautiainen, M. (2002). TREC-10 experiments at Maryland: CLIR and video. Proceedings TREC-10, NIST Publication #500-250, 549-561.

Fox, E.A. & Shaw, J.A. (1994). Combination of multiple searches. Proceedings TREC-2, NIST Publication #500-215, 243-249.

Harman, D. (1991). How effective is suffixing? *Journal of the American Society for Information Science,* 42(1), 7-15.

Kleinberg, J. (1998). Authoritative sources in a hyperlinked environment. Proceedings ACM-SIAM Symposium on Discrete Algorithms, 668-677.

Kraaij, W., Westerveld, T. & Hiemstra, D. (2002). The importance of prior probabilities for entry page search. Proceedings ACM-SIGIR'2002, 27-34.

Larkey, L.S., Ballesteros, L. & Connell, M.E. (2002). Improving stemming for Arabic information retrieval: Light stemming and co-occurrence analysis. Proceedings ACM-SIGIR'2002, 275-282.

Larkey, L.S. & Connell, M.E. (2002). Arabic information retrieval at UMass in TREC-10. Proceedings TREC-10, NIST Publication #500-250, 562-570.

Picard, J. (1998). Modeling and combining evidence provided by document relationships using PAS systems. Proceedings ACM-SIGIR'1998, 182-189.

Rasolofo. Y. & Savoy, J. (2003). Term proximity scoring for keyword-based retrieval systems. Proceedings ECIR-03, to appear.

Robertson, S.E., Walker, S. & Beaulieu, M. (2000). Experimentation as a way of life: Okapi at TREC. *Information Processing & Management*, 36(1), 95-108.

Savoy, J. (1996). Citation schemes in hypertext information retrieval. In *Information retrieval and hypertext*, M. Agosti, A. Smeaton (Eds), Kluwer, 99-120.

Savoy, J. & Picard, J. (2001). Retrieval effectiveness on the Web. *Information Processing & Management*, 37(4), 543-569.

Savoy, J. & Rasolofo, Y. (2001). Report on the TREC-9 experiment: Link-based retrieval and distributed collections. Proceedings TREC-9, NIST Publication #500-249, 579-588.

Savoy, J. (2002a). Report on CLEF-2001 experiments: Effective combined query-translation approach. In C. Peters, M. Brachler, J. Gonzalo, M. Kluck (Eds), *Cross-language information retrieval and evaluation*, CLEF 2001. Springer, Berlin, 27-43.

Savoy, J. (2002b). Report on CLEF-2002 experiments: Combining multiple sources of evidence. Proceedings CLEF-2002, 31-46.

Westerveld, T., Kraaij, W. & Hiemstra, D. (2002). Retrieving Web pages using content, links, URLs and anchors. Proceedings TREC-10, NIST Publication #500-250, 663-672.

Appendix 1. Weighting schemes

To assign an indexing weight w_{ij} reflecting the importance of each single-term T_j in a document D_i, we may use the formula shown in Table A.1, where document length (the number of indexing terms) for document D_i is denoted by nt_i, and n indicates the number of documents in the collection. For the Okapi weighting scheme, K represents the ratio between the length of document D_i measured by l_i (sum of tf_{ij}) and the collection's mean is noted by advl or more precisely

$$K = k_1 \cdot \left[(1-b) + b \cdot \frac{l_i}{avdl} \right]$$

For the Arabic corpus, the constant advl is set at 300, the constant b at 0.55, the constant k_1 at 3. For both Web searching tasks, we set advl at 750, the constant b at 0.9, the constant k_1 at 1.2. For the Lnu scheme, the constant pivot was fixed at 125 and the constant slope at 0.1.

bnn	$w_{ij} = 1$	nnn	$w_{ij} = tf_{ij}$
ltn	$w_{ij} = (\ln(tf_{ij}) + 1) \cdot idf_j$	atn	$w_{ij} = idf_j \cdot [0.5 + 0.5 \cdot tf_{ij} / \max tf_{i.}]$
lnc	$w_{ij} = \frac{\ln(tf_{ij}) + 1}{\sqrt{\sum_{k=1}^{t} \left((\ln(tf_{ik}) + 1) \right)^2}}$	npn	$w_{ij} = tf_{ij} \cdot \ln \left[(n - df_j) \big/ df_j \right]$
Okapi	$w_{ij} = ((k_1 + 1) \cdot tf_{ij}) \big/ (K + tf_{ij})$	dtn	$w_{ij} = (1 + \ln(1 + \ln(tf_{ij}))) \cdot idf_j$
ntc	$w_{ij} = \frac{tf_{ij} \cdot idf_j}{\sqrt{\sum_{k=1}^{t} (tf_{ik} \cdot idf_k)^2}}$	dtu	$w_{ij} = \frac{(1 + \ln(1 + \ln(tf_{ij}))) \cdot idf_j}{(1 - slope) \cdot pivot + slopc \cdot nt_i}$
ltc	$w_{ij} = (\ln(tf_{ij}) + 1) \cdot idf_j \Big/ \sqrt{\sum_{k=1}^{t} \left((\ln(tf_{ik}) + 1) \cdot idf_k \right)^2}$		
Lnu	$w_{ij} = \frac{\left(\ln(tf_{ij}) + 1 \Big/ \ln\left(l_i / nt_i \right) + 1 \right)}{(1 - slope) \cdot pivot + slope \cdot nt_i}$		

Table A.1: Weighting schemes

Classifier Stacking and Voting for Text Filtering

Rada MIHALCEA

University of North Texas
Denton, Texas, 76203-1366
rada@cs.unt.edu

Abstract

This paper summarizes the approach and the results of the TextCat system participating in the Filtering track in the Text Retrieval Conference 2002. The system relies primarily on statistical methods, and was designed with the main purpose of having a backbone system in which we can further integrate semantic components, and evaluate their relative performance as compared to traditional statistical approaches. The system is therefore simple, and is based on techniques for keywords extraction, and various classifier combinations including stacking and voting. TextCat participated in the *Batch* and *Routing* tasks. In the *Batch* task, it achieved a score of 39.02% normalized utility, and 26.37% F-measure respectively, averaged over all topics. The averaged uninterpolated precision for our best routing submission was 14.16%.

1. Introduction

The Filtering track has quite a long history in the Text Retrieval Conference (TREC) series. The goal of the track is to measure the ability of systems to classify new documents as relevant or irrelevant with respect to a given topic. While there are three different tasks organized within the Filtering track – adaptive, batch, and routing – our *Text Cat*egorization (TextCat) system participated only in the last two tasks. Few changes in the system would have probably allowed us to run TextCat on the adaptive filtering data as well; however, we decided to focus on the classification capabilities of the system, rather than on its adaptability to new incoming data. This is mainly because the purpose for building TextCat was to have a backbone text classification system, in which we can further integrate semantic modules and evaluate their relative performance as compared to the simple statistical approach. This follows up on our previous work in semantic-based Information Retrieval (Mihalcea, 2002), where various degrees of semantic knowledge where integrated into an existing Information Retrieval system (SMART (Salton and Lesk, 1971)). To extend this work to the text classification problem, we needed in the first place a basic text categorization system, which could then be expanded with more sophisticated modules. Since we were not able to find such a tool (reliable, free for download, with complete source code), we started building our own text categorization system, which ultimately resulted in the *UNT TextCat* system.

2. UNT TextCat

As stated in the title, TextCat relies on combinations of simple text classifiers, which includes stacking and voting. Starting with a basic *ngram-based classifier* and a *rule-based classifier*, we generate a range of new classifiers by making simple changes in the value of their input parameters. First, stacking is done by applying the rule-based classifier on the output produced by the ngram-based classifier. Second, classifier voting is performed using various degrees of inter-classifier agreement. In turn, different voting schemes generate new classifiers.

For the *Batch* task, we had a total of thirteen stacked classifiers, which were then fed to the voting scheme, such that ten more combined classifiers were generated. Out of this total number of 23 classifiers, one was chosen according to its performance during cross validation runs performed on the training data. This tuning on training data was done separately for the normalized utility measure and for the F-measure, resulting in two different submissions, *UNTextCatSU* (run optimized for the T11SU measure), and *UNTextCatF* (run optimized for the T11F measure).

For the *Routing* task, we used a single combination of the thirteen stacked classifiers (run *UNTextCatR*), and a combination of the thirteen ngram-based classifiers, with no prior stacking (run *UN-TextCatR1*)

2.1. Data

This year, the text collection for the Filtering track consisted in the Reuters documents published during August 1996 - August 1997. To speed-up the classification process, and avoid the overhead associated with the repetition of some initial transformation procedures, there is a pre-processing phase where all documents are transformed and saved in a format suitable for the text classifiers. During this phase, words are stemmed using Porter stemmer (Porter, 1980), and common words are eliminated based on a list of about five hundreds words that comes with the SMART system package. The output of this stage is a single large training file that includes all training documents, one document per line, each line being preceded by the document identifier. Similarly, there is one large test file that includes all test documents. Even though the pre-processing procedure was designed for the specific format of Reuters documents, we expect to be able to easily adapt it to new data formats, with a minimal number of changes.

2.2. Ngram-Based Classifier

The first classifier consists of a simple ngram-based classification scheme. Specifically, we have a module that generates candidate ngram keywords starting with the training files. So far, the candidate keywords consisted only of unigrams and bigrams. In future work, we plan to investigate the impact of longer ngrams on the quality and efficiency of the TextCat classifier.

Shortly, the ngram-based classifier proceeds as follows. First, we select an initial large set of ngrams from the training files, based on their frequency in the documents considered relevant for the given topic. Next, for each such ngram, we calculate several parameters, including frequency in each relevant document, ratio between relevant and irrelevant documents that can be extracted with the ngram, correlation coefficient, number of relevant documents that include the ngram. A fixed number of ngram-*keywords* is selected from this large list, based on their relative value with respect to some threshold values set for their parameters. Finally, if more than a certain number of keywords fulfill the minimum requirements, the highest ranked ngrams are selected. This list of ngram-keywords is then adjusted through several loops, where the number of keywords may be increased or decreased, based on the total number of test documents that are extracted. This is based on the intuition that a fixed number of keywords may not be satisfactory for all topics. An efficient classification for a certain topic may be performed with only 5 keywords, whereas another topic may need as many as 15 keywords or even more. We also set a maximum over the number of loops that may be executed, to avoid excessive running times for certain topics. Finally, once the list of ngram-keywords is selected, all documents from the test set that contain any of these keyword are classified as relevant, and all remaining documents are classified as irrelevant.

There are several parameters that may influence the number and the ranking of the ngram keywords. Consequently, the settings made for these parameters may also influence the number of documents classified as relevant or irrelevant to the given topic. In TREC 2002, TextCat included thirteen different settings, which therefore resulted in thirteen different classifiers. The list below details the various parameters that may be set for the ngram-based classifier.

TOP_NGRAMS Maximum number of ngrams pre-selected from the training files. Final set of keywords is selected from this preliminary list.
Default value: 200

RATIO_RELEVANT_IRRELEVANT The ratio between relevant and irrelevant documents in the set of documents used for ngrams pre-selection.
Default value: 512

TOP_NGRAM_KEYWORDS Initial number of ngram-keywords that is extracted from the preliminary list of ngrams. This number is subsequently adjusted through several loops.
Default value: 10

MIN_FREQUENCY The minimum value acceptable for the frequency of a ngram in each relevant training document, for the ngram to be selected as a keyword.
Default value: 3

MIN_DOCSREL The minimum number of relevant documents that should include the ngram, such that the ngram is selected as a keyword.
Default value: 2

MIN_CORRELATION The minimum value acceptable for the correlation coefficient associated with each ngram, such that the ngram is selected as a keyword. The correlation coefficient was defined in (Ng et al., 1997):

$$\frac{(N_{r+}N_{n-} - N_{r-}N_{n+})\sqrt{N}}{\sqrt{(N_{r+} + N_{r-})(N_{n+} + N_{n-})(N_{r+} + N_{n+})(N_{r-} + N_{n-})}} \quad (1)$$

where N_{r+} (N_{n+}) is the number of relevant (irrelevant) documents containing the given ngram, and N_{r-} (N_{n-}) is the number of relevant (irrelevant) documents that do not contain the given ngram.
Default value: 0.02

MIN_DOCS Minimum number of documents to be extracted from the test files. This is the lower bound in the loop that selects keywords. If the number of documents extracted is lower than this threshold, than the number of selected keywords is increased, so that additional documents are extracted from the test set.
Default value: 1000

MAX_DOCS Maximum number of documents to be extracted from the test files. This is the upper bound in the loop that selects keywords. If the number of documents extracted is larger than this threshold, than the number of selected keywords is decreased, so that fewer documents are extracted from the test set.
Default value: 2000

MAX_LOOPS Maximum number of loops that can be executed for keyword selection. Starting with the initial set of TOP_NGRAM_KEYWORDS, keywords are added or removed from this set, until the number of documents that are extracted falls in the range of MIN_DOCS - MAX_DOCS. This loop will stop after is executed for MAX_LOOPS times, regardless of the number of documents retrieved.
Default value: 10

2.3. Rule-Based Classifier

The rule-based classifier used by TextCat is Ripper, an off the shelf system available from AT&T (Cohen, 1995). The reason for choosing Ripper as our rule-based classifier was twofold. First, it was previously shown that Ripper is an efficient classification scheme for text categorization problems (Cohen and Singer, 1996). Second, Ripper handles set-valued features, and therefore we can feed the entire document as a single attribute, and let the machine learning algorithm decide upon the contribution of various keywords for the relevance classification.

2.4. Classifiers Stacking

The first method that we employ for classifier combination is stacking, where the rule-based classifier is applied on the output produced by the ngram-based classifier. The documents considered relevant by the ngram-based classifier are therefore the only documents seen and classified by the rule-based learner. This stacking procedure is also meant as a speed-up in the classification of large text collections, since the rule-based learner does not handle very well collections that exceed a certain size. Systems participating in the filtering task had to deal with collections of over 800,000 documents, and consequently the single use of the rule-based learner was not a feasible solution.

2.5. Classifiers Voting

Besides stacking, additional classifier combinations are performed through a simple voting scheme, where the number of inter-classifier agreements is collected, starting with a set of thirteen base classifiers. Basically, for each document that is considered relevant by any of these base classifiers, we count the number of votes the document receives from the remaining classifiers. Next, a minimum acceptable value is set for this vote, and only those documents that have a total vote exceeding the given threshold remain in the final classification. There are ten different threshold values considered in the TREC 2002 experiments, ranging from 0 (meaning that at least one base classifier should find a document to be relevant, for the document to be considered relevant in the final classification) to 10 (meaning that a document is classified as relevant only if at least ten of the base classifiers give a relevance vote to that particular document). These ten different threshold values resulted in ten additional classifiers. Hence, the final number of classifiers used in this task was 23, that is thirteen base classifiers, plus ten combined classifiers.

3. UNT TextCat at TREC 2002

In TREC 2002, TextCat participated in the *Batch* and *Routing*. All filtering systems were evaluated based on: (1) normalized utility measure, which relates the relevant and irrelevant documents in the set of retrieved documents (N_{r+} and N_{n+}); the normalized utility is intended as a measure of the number of irrelevant documents that a user can tolerate in a given set of retrieved documents; (2) the F-measure, which is a standard measure in Information Retrieval (Van Rijsbergen, 1979), and combines the precision and recall figures obtained for a certain topic. Precision and recall were also measured individually for each topic.

In the *Batch* task, TextCat achieved a score of 39.02% normalized utility, and 26.37% F-measure, averaged over all topics. The normalized precision for our best routing submission was 14.16%.

3.1. The *Batch* task

In the *Batch* task, thirteen base classifiers are built by varying the relative ratio between relevant and irrelevant training documents. The first classifier uses all training documents provided in the *QRels* judgment file, with their corresponding relevance judgments. The second classifier adds to this initial set of documents an equal share of irrelevant documents. The third, and all subsequent classifiers, double each time the number or irrelevant documents, up to the thirteen classifier, which trains on all N documents listed in the *QRels file*, plus an additional number of 2048*N irrelevant documents. As a rule of thumb, classifier i includes all N documents listed in the *QRels* file, plus an additional set of $2^{i-1} * N$ irrelevant documents, all of them extracted from the training collection. If not enough irrelevant documents are found in the training set, the ratio 2^{i-1} reflects a maximum, rather than the actual rapport between relevant and irrelevant documents. In all these classifiers, all parameters except the RATIO_RELEVANT_IRRELEVANT were used with their default values. New classifiers can be easily generated by changing the values of these input parameters.

Next, the rule-based classifier is combined with each of these initial classifiers through stacking. As noted earlier, ten additional classifiers are built by combining the thirteen base classifiers using a voting scheme.

To select one classifier out of the total set of 23 different classifiers, cross validation runs were performed on the initial training set, with the ratio between training and test documents set to 90%-10%. There were two TextCat submissions in the *Batch* task, one optimized for the utility measure – *UNTextCatSU* – where the classifier leading to the highest normalized utility score during cross validation runs is the one that is selected, and one optimized for the F-measure – *UNTextCatF*.

Table 1 lists the thirteen base classifiers and the ten combined classifiers, with their corresponding average utility and F-measure, as well as the averaged precision and recall. Moreover, the last two columns list the number of times each classifier was selected in the cross validation phase, for each run (optimized for normalized utility or for F-measure).

As seen in the last two columns in Table 1, the classifier selection is quite uniformly distributed among the 23 different classifiers; however, base classifiers one and thirteen seem to be selected more often, as compared with the selection frequency for the other classifiers. In terms of performance, base classifier thirteen has an utility measure of 33.87%, which is 5% smaller than the score for the final classifier; in real world applications, this relatively small difference may not fully justify the additional running time brought by the cross validation phase, and therefore in some applications this could be the only classifier employed for the filtering task. The maximum F-measure that can be achieved with a single classifier (base or combined, with no selection) is 18.28%.

Figure 3.1. plots the scores obtained by all classifiers for the four different measures. For the base classifiers, a steady growing tendency is observed for the utility measure, which suggests that the larger the

					Times selected	
Classifier	T11SU	T11F	Prec.	Recall	(opt.SU)	(opt.F)
Base classifiers						
1 (N)	2.01%	2.13%	2.28%	29.53%	17	12
2 (N + N irrel.)	16.00%	7.34%	10.14%	13.05%	2	1
3 (N + 2N irrel.)	17.96%	10.26%	14.31%	13.06%	3	3
4 (N + 4N irrel.)	17.38%	12.30%	16.06%	14.25%	4	6
5 (N + 8N irrel.)	20.29%	12.62%	16.65%	13.52%	4	5
6 (N + 16N irrel.)	21.03%	13.92%	19.37%	14.33%	2	4
7 (N + 32N irrel.)	21.79%	15.02%	19.97%	14.12%	3	4
8 (N + 64N irrel.)	24.13%	15.58%	20.77%	13.86%	4	5
9 (N + 128N irrel.)	24.94%	15.89%	20.17%	14.25%	5	3
10 (N + 256N irrel.)	25.80%	16.00%	22.98%	13.35%	6	9
11 (N + 512N irrel.)	30.50%	18.08%	25.41%	13.93%	4	6
12 (N + 1024N irrel.)	31.89%	16.02%	23.12%	10.98%	0	2
13 (N + 2048N irrel.)	33.87%	14.82%	22.95%	9.89%	19	12
Combined classifiers						
1 (min.1 vote)	4.24%	7.42%	6.76%	34.93%	0	0
2 (min.2 votes)	13.47%	13.88%	14.50%	26.15%	2	1
3 (min.3 votes)	19.13%	16.47%	19.17%	22.14%	2	0
4 (min.4 votes)	24.45%	18.28%	24.30%	18.04%	0	0
5 (min.5 votes)	25.78%	17.12%	26.19%	14.01%	3	3
6 (min.6 votes)	25.80%	16.56%	27.12%	11.78%	4	6
7 (min.7 votes)	23.00%	14.91%	27.31%	9.93%	3	4
8 (min.8 votes)	20.46%	12.66%	26.83%	8.03%	4	2
9 (min.9 votes)	16.19%	9.64%	23.57%	6.27%	4	2
10 (min.10 votes)	13.95%	8.68%	22.52%	4.34%	5	3

Table 1: Individual results obtained with each base/combined classifier

number of irrelevant documents, the better. The recall is relatively constant across different base classifiers, with the only exception being classifier one, which has a significantly higher recall. Precision and F-measure achieve a maximum for base classifier eleven, followed by a decrease in classifiers twelve and thirteen. In the case of combined classifiers, there is a peak in utility, F-measure, and precision, for combined classifiers four through six, followed again by a sharp decrease. As expected, the highest recall is obtained with combined classifier one. What this figure suggests is that new system settings may eventually lead to even higher scores for various measures (e.g. larger number of irrelevant documents for higher utility score, larger number of base classifiers in the voting scheme for higher recall, etc.).

3.2. The *Routing* task

In the *Routing* task, two different TextCat runs were submitted. The first submission, *UNTextCatR*, consists in the top 1000 documents returned by the combined stacked classifier with a minimum vote of one (i.e. combined classifier one). The second submission, *UNTextCatR1*, consists in the top 1000 documents returned by a combined classifier, again with a minimum vote of one, but this time with no prior stacking (that is, only the ngram-based classifier is employed during this run). The average normalized precision was 14.16% for *UNTextCatR*, and 8.15% for *UNTextCatR1*.

4. Conclusions

This paper has described the approach and the results obtained with TextCat – a simple text filtering system that relies on various classifier combination

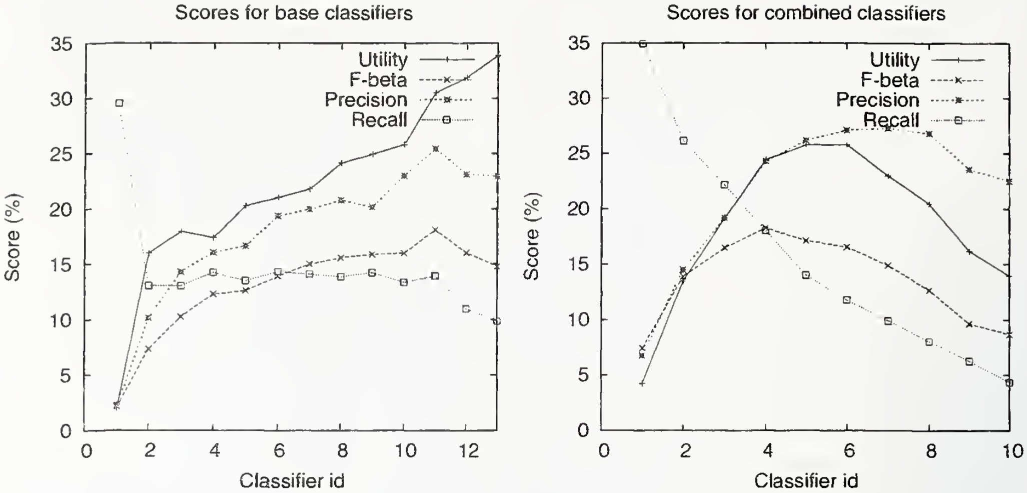


Figure 1: Utility, F-measure, Precision and Recall obtained with various classifiers

schemes. In the *Batch* task, the average utility score achieved with TextCat was 39.02%, and the average F-measure was 26.37%. In the *Routing* task, the two TextCat submissions achieved an average uninterpolated precision of 14.16% and 8.15% respectively. As a next step, we plan to integrate semantic modules into TextCat, and evaluate their relative performance as compared to the simple statistical-based approach.

5. References

W. Cohen and Y. Singer. 1996. Context-sensitive learning methods for text categorization. In *Proceedings of the 19th Annual International ACM SIGIR, Conference on Research and Development in Information Retrieval*, pages 307–315, Zurich, CH, July.

W. Cohen. 1995. Fast effective rule induction. In *International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, July.

R. Mihalcea. 2002. Going beyond explicit knowledge for improved semantic based information retrieval. *International Journal on Tools with Artificial Intelligence*, 11(4), December.

H.T. Ng, W.B. Goh, and K.L. Low. 1997. Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Philadelphia, PA, July.

M. Porter. 1980. An algorithm for suffix stripping. *Program*, 14(3):130–137.

G. Salton and M.E. Lesk, 1971. *Computer evaluation of indexing and text processing*, pages 143–180. Prentice Hall, Ing. Englewood Cliffs, New Jersey.

C.J. Van Rijsbergen. 1979. *Information Retrieval.* London: Butterworths. available on-line at http://www.dcs.gla.ac.uk/ Keith/Preface.html.

PiQASso 2002

G. Attardi A. Cisternino F. Formica M. Simi A. Tommasi*

{attardi, cisterni, formicaf, simi, tommasi}@di.unipi.it
Dipartimento di Informatica
University of Pisa, Italy

Abstract

The University of Pisa participated to TREC 2002's QA track with PiQASso, a vertical QA system developed (except for some of the linguistic tools), entirely within our research group at the Computer Science department. The system features a filter-and-loop architecture in which non-promising paragraphs are ruled out basing on features ranging from keyword matching to complex semantic relation matching. The system also exploits the Web in order to get "hints" at what to look for in the internal collection. This article describes the system in its entire architecture, concentrating on the Web exploitation, providing figures of its efficacy.

1 Introduction

Last year's PiQASso system featured an architecture similar to that deployed by many systems [1]. Such architecture is organized in a series of subsequent filters that rule out paragraphs that are evaluated as not answering the question.

We kept preprocessing on the collection to a minimum, basing on the following considerations:

- pre-processing the whole collection is an expensive process;
- ideally, the collection to use as knowledge base to mine answers should be as large as possible: this makes systems that perform extensive processing on the data collection less scalable;
- the collection might even be changing dynamically, which in fact renders such systems unusable under this assumption;
- in an experimental setup, several different algorithms might be experimented, and re-processing the whole collection for each experiment is unfeasible;
- not assuming any preprocessing on the collection allows for immediate application to "foreign" collection, e.g. by querying a search engine.

Whenever the following holds:

- the collection is static;
- the set of operations to perform on it is "stable";
- the implementation of such operations is stable as well;
- the "enriched" collection size does not exceed disk space;

nothing prevents from performing all the necessary computing on the collection off-line, obtaining substantial speed-up.

As with most other systems, analysis of the question is performed as the first step during the question answering phase. Within PiQASso, the goals of such analysis are:

- identify the suitable keywords to perform IR-style search;
- classify the question onto the expected answer type (EAT) taxonomy;
- extract a set of "logical" relations connecting the question's words;

*Alessandro Tommasi would like to thank Microsoft Research, Cambridge, for supporting his PhD.

- assign a weight to each word, proportional to the "importance" of the word.

Each of these 4 characteristics of the question are used to rule out sentences from the set of candidate answers.

The overall system architecture can be sketched as follows. Each question undergoes question analysis. From the analysis, data is gathered to apply the following series of cascade filters to the collection:

retrieval IR performed by the keywords extracted from the query, and in proximity, returning paragraphs;

type filter paragraphs not containing entities of the expected answer type are discarded;

relation matching sets of logical relations between words are extracted from the paragraph and the question and a matching cost is computed deploying a measure based on *semantic distance* between words;

popularity ranking whenever multiple evidences of the same answer are collected, this boosts its confidence score.

If no answer makes it through this cascade of filters, the retrieval phase is re-executed by loosening the search, in a loop.

This article describes each of these steps. In addition it describes the use of the Web to narrow the IR phase, along with the results obtained over TREC 2002's question set.

2 Tools deployed

2.1 Indexing

This year's task was based on a different collection than 2001's: the AQUAINT collection. We indexed the collection similarly to last year: we deployed IXE [2], an efficient indexing and search engine based on template meta-programming and developed within our group[1], to build full-text indexes of the collection. Along with the full-text, we have stored paragraph splitting information, obtained by the processing of a Maximum Entropy-based sentence splitter. This way, by customizing IXE, we are able to issue queries and have the search engine to return results as paragraphs rather than documents. However, the indexes are still built over documents (i.e., we did not consider each paragraph as a separate document searchwise), allowing for proximity search to span over different adjacent paragraphs.

Just as it did for paragraph splitting information, IXE is able to store and handle additional information about the text being indexed. We are foreseeing moving part of the "inevitable" (e.g., NE tagging) work to the index phase, so that it could be exploited by issuing queries in a richer language (for instance, specifying the type of the entity sought for).

2.2 Parsing

To perform syntactic parsing of sentences we used Minipar [5], a dependency parser developed by D. Lin and free for non-commercial use. The parser has several additional capabilities that come in very handy for our task; in particular:

- it identifies logical relationships (subject, object, complement) rather than grammatical ones;
- it tags words with semantic tags, providing a form of named entity recognition;
- it partially solves coreference resolution problems;
- the dependency tree it returns turned out to be a good way of representing text.

In figure 1 we see, for each word, linked to the head word, its relation to the father, its label, the part of speech, the morphological root, the semantic tags (in braces). Also, note, for the word "which", the "@5": it means that the pronoun refers to the word of label 5: "Messiah". Some of the tags (for instance, the "quote" tag) have been added by us to recognize particular entities, in this case quoted entities. For robustness, we have also added part of speech tags via an external part of speech tagger (TreeTagger [10]), in order to compare Minipar's output with TreeTagger's. This way we have greater confidence about the POS tag when both tools agree on it.

2.3 Named entity recognition

Just as we used an external POS tagger to verify Minipar's tagging, we also used an external NE-Tagger to

[1] Research supported by Ideare s.r.l.

```
_EMPTY ((null)  E2  (null)  (null)   (null) )
+--_EMPTY ((null)  E1  C  fin   (null) )
|     +--composed (i  2  V  compose   VBN )
|     |     +--Handel (s  1  N  Handel  {person} NN )
|     |     +--_EMPTY (subj  E3  N  Handel  {person}@1 (null) )
|     |     +--Messiah (obj  5  N  the Messiah  {quote} NP )
|     |     |     +--the (lex-mod  4  (null)  the  {quote} DT )
|     |     |     +--in (mod  7  Prep  in   IN )
|     |     |     |     +--1741 (pcomp-n  8  N  1741  {date} CD )
|     |     |     +--_EMPTY (rel  E0  C  fin   (null) )
|     |     |     |     +--which (whn  10  N  which  @5 WDT )
|     |     |     |     +--rearranged (i  13  V  rearrange   VBN )
|     |     |     |     |     +--was (be  11  be  be   VBD )
|     |     |     |     |     +--later (amod  12  A  later   RB )
|     |     |     |     |     +--_EMPTY (obj  E4  N  which  @5 (null) )
|     |     |     |     |     +--by (by-subj  14  Prep  by   IN )
|     |     |     |     |     |     +--Mozart (pcomp-n  15  N  Mozart  {person} NP )
```

Figure 1: **the tree returned by Minipar for the sentence:** ***Handel wrote "the Messiah" in 1741, which was later rearranged by Mozart.***

provide additional information about entities that Minipar did not recognize. The results of the NE-Tagger simply "enrich" the annotation provided by Minipar.

The NE-Tagger we chose is based on Maximum Entropy [9], whose accurary depends mostly on the quality of the training set used. The one we used is small and originally designed for speech recognition tasks. Also because of this, the entity taxonomy is small: entities can be: *person, organization, location, money, measure, cardinal, percent, duration, date, time.* These categories are however the standard ones as defined in the relative MUC task [3].

A mapping among the different taxonomies of the various tools (Minipar, the NE-Tagger and WordNet) is required.

2.4 WordNet

WordNet [8] has been used mainly for: classifying words (to determine the EAT) and measure the *semantic distance* of two words. This second activity has the goal of returning a score that is higher if the two words are semantically very close. This "closeness" refers to the interchangeability of the two words in a sentence. Consider for example the word "cat". The word "mammal" is definitely not a synonym of cat, yet it's a closer term, semantically, than "lizard". While this is obviously a context dependent property, it has been globally approximated.

The type of a word is simply the taxonomy the term belongs to according to WordNet. This yields to 24 categories, some of which can be mapped directly onto Minipar or the NE-Tagger categories, and others that constitute categories by themselves.

In order to evaluate the semantic distance or to chose the proper type of a word, the right *synset* (sense) of a word must be selected. This is, again, a property that should take the context into account. However, for several practical reasons (for instance, because the question provides too little a context), we have approximated it more pragmatically, exploiting WordNet's sorting of senses by frequency. For type identification, we return for word w, whose ordered senses in WordNet are $\{s_0, \ldots, s_n\}$, the category C that maximizes:

$$P(w,C) = k \sum_{j=0}^{n} \frac{n-j}{n} \gamma(s_j)$$

where

$$\gamma(s_j) = \begin{cases} 1 & \texttt{iff} \quad s_j \in C \\ 0 & \texttt{otherwise} \end{cases}$$

and k is a constant, presently set to 0.7.

For semantic similarity the various senses of the two words are weighted, and contribute differently to the final score. Word similarity, is computed over nouns, verbs, adjectives and adverbs, while only the first two categories were treated last year.

3 Question analysis

The question is parsed. From then on, all further analysis takes place on the parse tree, which is possibly enriched by means of other tools.

3.1 Keyword extraction

The first important thing done on the question is the selection of those terms in the question that should be part of the IR-like query. A word is selected as a keyword if:

- it is not a "forbidden" word like wh-words;
- if it is an informative noun (you, me, something etc. are non-informative);
- it's a verb but it is not "to be";
- it is an adjective or an adverb.

The keyword set can be modified in 4 successive, progressively loosening steps in case no answer is found. First, morphological variations for each word are added. Second, synonyms (according to WordNet) are `OR`'ed to each keyword. Third, adverbs and first proper names are dropped. Fourth, verbs are dropped as well, and if more than 3 keywords are still there, all keywords whose father (according to the parse tree) is a keyword are dropped. The empirical justification is that in a dependency parse tree, the children of a node "modify" the node, that generally has a more central role in the sentence (the same rationale drove the word weight assignment, see 3.3).

3.2 Expected Answer Type

The EAT is the type of the entity that is asked by the question. The taxonomy for the EAT is not very uniform, for it reflects the natural taxonomy of questions. It is composed of the categories: `person`, `organization`, `location`, `time-date`, `quantity`, `quoted`, `language`, and those gathered from WordNet. The `person` category also features two subclasses, depending on whether the question expects a proper noun or rather a definition. The EAT can also be `person OR organization`, for questions of the kind "who did something". The category `quoted` is meant for titles (of books, songs, movies...), while the category `language` is for questions of the kind "what does something mean/stand for". The remaining categories (those referring to WordNet's taxonomy) are for the "what" kind of question, that usually provide a narrower description of the entity they're asking for by a focus noun such as "instrument" in "what instrument did Glenn Miller play?".

3.3 Weights

In our abstraction, words in the question express relations (verbs), attributes (adjectives and adverbs), concepts (common nouns) or entities (proper nouns) for which we expect to find a counterpart in an answering paragraph. However, not all the words in the question are expected to be found in the answer paragraph, nor all of them are sought with the same priority. Therefore, we assign a *weight* to each word in the question. This weight represents the relative importance of the concept associated with the word, and will influence the matching cost between question and answer by which candidate answers are sorted.

The weight is a number in the [0, 1] interval. Words that are not initially selected as keywords all have weight 0 (they may happily be missing from the answering paragraph). For all others, we visit the parse tree recursively. The first word has weight 1, and the weight halves as the depth of the node increases. However, nodes that are tagged with semantic features receive a 10% "bonus" weight for each feature they have.

The rationale of this heuristic is that nodes closer to the root (the main verb and its direct complements) express more "basic" properties of the required entity, while down deeper in the dependency tree are only additional, perhaps "optional" requirements that are less likely to appear in an answering paragraph.

4 Answer Selection

4.1 Retrieval: Proximity Search

We combine the keywords extracted from the question in a proximity query. The width of the proximity window is roughly estimated as being twice the number of nodes in the parse tree of the question. All paragraphs over which the resulting text window spans are retrieved as candidate answers. This allows for "elasticity": if a particular paragraph is retrieved, but it does not contain all the question keywords, we can rest assured that together with its close context (surrounding paragraphs) it did.

All paragraphs retrieved this way are however considered independently.

4.2 Type Filter

Of all the paragraphs returned by the retrieval phase, the top N (we have experimented with N's ranging from 400 to 1200) are considered. These paragraphs are parsed for further analysis, the first step of which is the type filter. The parse tree is visited, looking for at least a node that is tagged (or that is classifiable) in the EAT category of the question. Furthermore, it is required that such node did not occur in the question as well. Consider the question: "Who killed John Kennedy?", whose EAT is *person* OR *organization*. All paragraphs containing John Kennedy (and there will be plenty of these in the search result) contain a *person* node; for sure we are looking for a different person.

Paragraphs that do not contain an entity of the proper type are discarded.

4.3 Relation extraction

Once the candidate paragraph has passed the type filter, we want to make sure the paragraph actually *supports* its answer. For factual questions like TREC's, an answer is an entity that verifies the conditions imposed by the question. We have issued the following hypotheses:

- the conditions imposed by the question are expressed by the logical relations among words;
- such relations are found in answering paragraphs.

While verbs are usually considered as relations among entities expressed by nouns, we have reified them, and considered them as "concepts" too. This way, the number of relations is considerably cut down to syntactical ones, like "subj, obj, p-comp" etc., as output by Minipar.

To achieve better flexibility with respect to the various different expressions of the same concepts, we must perform syntactical normalization. John's ownership of a car is expressed by both "John has a car" and "The car of John". To make sure we are able to understand that sentences like these convey the same meaning, we need to perform normalization, for which we have flattened the dependency tree to a set of relations (head word, relation, modifier word), which is less strict, and allows us to insert new relations without messing up the structure.

First, all relations found in Minipar are inserted in the relation set. From then on, a rule system is applied to the set, extending it, until the fix-point is reached (no more relations are added). Rules can be very general, but are usually in the form: "if A is in relation R_1 to B, and B is in relation R_2 to C, then add the relation (A, R_3, C)". With respect to the example above, a rule for "to have" says that: "if A is the subject of the verb to have, and C is the object, then A is a specifier of C". This way, both example text snippets would "agree" on the relation (C, specifier, A).

In the current system, we have 19 such rules, all oriented to syntactical normalization.

In a domain as open as TREC's, it is hard to imagine how semantic rules of the same kind could be added. However, in principle nothing keeps from adding more rules for specific domains. In our current system, only three semantic rules have been added, mainly for testing purposes: "if A is known as B, then A is B"; "if A directed, composed, wrote, manufactured, produced or invented B, then it A is a specifier of B", "if A means/stands for B, than A is B". These rules are easy to write, and all they do is enriching the relation set relative to a paragraph. However, it is impractical to think we could write rules for all common sense facts.

Relation sets are extracted both from candidate paragraphs and from the the question, for which we also set an empty slot, a node that is missing in the question, and yet it's in relation with question words. This empty slot represent the answering entity: the node with similar relations to this one in the answering paragraph is considered to be the had of the answer. The position of the empty slot in the parse tree (and therefore the relations it has with other words) is usually the position of the first "missing" node; we look for the first verb whose subject or object is missing. Special cases correct "when" and "where" questions, for which such node is placed as a modifier of the main verb.

4.4 Relation matching

Once the relation sets for question and candidate answer are determined, we use a matching function to determine how well an entity in the paragraph matches the

Q: *Who was the first person to reach the south pole?*
A: *Any bright schoolchild can tell you that Roald Amundsen was the first man to reach the South Pole, but can anybody name the first baseball fan to brave Candlestick?*

`reach, Amundsen (subj) <-> reach, ANODE ((null))`
`VROOT, first (post) <-> VROOT first (lex-mod)`
`reach, Pole (obj) <-> reach, pole (obj)`
`Pole, South (lex-mod) <-> pole, south (lex-mod)`

Figure 2: **an interesting case, in which the answer is provided by a question. We show the matching relations (on the left of** `<->, the answer relations,` **on the right, the question ones). ANODE is the empty slot in the question. Notice how the subject of "reach" in the answer is Amundsen, while the word "first" cannot be attached to it, and is therefore "pending" (attached to a virtual root VROOT).**

question's requirements, allowing for sorting answering paragraphs. All paragraphs that are more distant than a threshold are considered not answering.

The matching algorithm begins by matching the question's empty slot against a node of the required type in the answer (*answering node*). The set of nodes in relation with the empty slot is then identified. For each such node, a node in relation with the answering node is looked for as "partner". The algorithm then proceeds by trying to match nodes of distance 2 from the empty slot to nodes of distance 2 from the answering node, and so on. We find the optimal solution, choosing for each node in the question a partner in the answer so that the global matching cost is minimal.

The global matching cost is defined in terms of several properties.

- Nodes in the question that do not have a match in the answer contribute to the cost proportionally to their weight (question requirements missing in the answer);

- For each question node/answer node association, a contribution to the cost is added proportionally to the weight of the question node and the distance of it from the answer node.

The distance from a question node to the corresponding answer node is evaluated basing on the similarity measure defined on WordNet, and on other properties, such as:

- the number of semantic features "missing" in the answer node;

- the difference of modifiers;

- different relation with the father.

An example of the matching returned for a question and a candidate answer is shown in figure 2, in which the matching is more compactly shown as associations between relations extracted from the question and the answer.

5 Exploiting the Web

5.1 Narrowing the search for the answer

While the type filter and the relation matching phase are highly semantic filters, the most selective, the one that reduces the number of paragraphs from the millions to a few hundreds, is the IR phase. If questions like "What is an atom?" leave little choice as to what to deploy as keywords, in several other cases, for more complex questions, the choice of keywords plays a crucial role. Several times, morphological variations or adding synonyms does not help: a honeymoon implies a wedding, but it is not a synonym.

The difficulty of finding the proper paragraphs for complex questions is evident from figure 3, in which it is shown that the system performs considerably better for short questions. Moreover, keeping at most the top 400 results of every query, over the 500 TREC2002 questions we retrieved in total about 180.000 paragraphs of which only about 1 every 30 matched the corresponding answer string (lists of regular expressions matching correct answers to each question, and kindly provided by Ken Litkowski [6] to the QA-track mailing list).

Figure 4 shows that the number of questions for which no paragraph retrieved contained an answer is indeed high.

Average words in question: 6.3	
Longer questions: 215	Shorter questions: 285
Correct: 44	Correct: 123

Figure 3: **a comparison between "long" and "short" questions. Short ones appear to be easier, mainly because for long ones, it is more difficult to select the right keywords (TREC 2001 question set, our best submitted run). The TREC 2002 question set amplifies this problem, for the average length of a question increased to 7.6 words per question, still with 285 questions under the average.**

	# questions	# correct NILs
0 par. returned	37	14
0 ans. within par.	180	37
1+ ans. within par.	173	//

Figure 4: **numer of questions for which: no paragraphs were returned at all, some paragraphs were returned but none of them contained an answer matching Litkowski's patterns, some paragraphs matching the patterns was retrieved. On the right column, the number of questions for which the correct answer was NIL (TREC 2002 set).**

Hypothesis: the Web features such a high redundancy, that it is likely to find a fact stated in a particular way.

Assuming this fact, we have decided to issue a very narrow query to a search engine, to see if an answer could be found on the Web. Such answer (that we call "Web suggestion") can then be searched for directly on the internal collection

5.2 Answer template

During the question analysis, an *answer template* is constructed: a text fragment obtained by turning the question into a direct form. Examples of pairs of question and answer template are provided in figure 5.

Templates are generated simply by turning the main verb into the direct form, and adjusting its conjugation (did originate → originated, does have → has).

Additionally, looser templates are obtained from these ones by simply removing verbs (and therefore breaking the template in two or more chunks). This is done in case no suggestion is found on the Web using the stricter template.

5.3 Google candidates

By means of Google's API, we issue the answer template as query to Google, and isolate the resulting excerpts. By the way Google works, such excerpts contain the text in the document retrieved that matched the query. In that respect, it is not dissimilar from the paragraphs we return by our search engine. Also, because the query asks for adjacent words, we know that the relative excerpts must contain a text fragment that is (almost) well constructed, and therefore parsable.

Some examples of excerpts are shown in figure 6.

We apply type filtering and relation matching to these excerpts, in order to obtain actual verified answers out of the excerpts. Because the search string was the question turned into direct form, it is usually easier to verify its correctness.

5.4 Justifying Web suggestions in the collection

Even if the answers gathered from the Web are justified by their exceprt, it is the case that such excerpts might indeed be wrong, or that an answer must be found on an internal, trusted collection.

To do so, and bearing in mind that several irrelevant paragraphs were returned by the "standard" search strategy, we have "doped" the keyword set by adding the answers found on the Web. This in fact renders the search much narrower, and hopefully, more likely to find answers. In fact, we are searching the question's keywords along with answers to the same question as found elsewhere.

In figure 7 some question/suggestions pairs are shown. From the figure we see that these candidates have a high rate of correctness, mainly due to the strict template they were gathered from, and to the answer validation that is applied to them. In fact, while the template is often very good by itself in selecting good answers, validation can rule out cases such as "Galileo was very nice", or "Wilt Chamberlain scored 100 points and then showered", which match the template but do not contain answers.

In the same figure we see another fact that is important to consider when looking for answers in the Web.

Question	Template
In what country did the game of croquet originate?	The-game-of-croquet-originated
What year did Wilt Chamberlain score 100 points?	Wilt-Chamberlain-scored-100-points
How many chromosomes does a human zygote have?	A-human-zigote-has
What lays blue eggs?	Lays-blue-eggs
Which vintage rock and roll singer was known as "The Killer"?	Was-known-as-The-Killer

Figure 5: **some questions with the relative answer template. A hyphen between words means that they must be adjacent (Google syntax).**

Google Excerpt
... AP. The following is The Associated Press story filed March 2, 1962, the night Wilt Chamberlain scored 100 points in a game against the New York Knicks. ...
... March 2 came and went quietly, but was a noteworthy date in sports history ? Saturday was the 40th anniversary of the day Wilt Chamberlain scored 100 points ...
... 28. The ball was supposedly the one that 7'1" Wilt Chamberlain scored 100 points with on March 2, 1962, in Hershey, Pennsylvania. ...
... The Associated Press. After Wilt Chamberlain scored 100 points in a game in Hershey, Penn., Kerry Ryman snatched the ball after the contest. ...

Figure 6: **some of the excerpts returned in the first result page of Google for the query "Wilt-Chamberlain-scored-100-points".**

Question	Web Suggestions
Who was Galileo?	astronomer, scientist, philosopher, son, physicist, professor, pioneer, egoist
What is an atom?	quantum, block, Raspberry, Lies, part, Friend, thing, hero, particle, helium, Port, member, vehicle, constituent, piece, organization, proof
What year did Wilt Chamberlain score 100 points?	1962, 1984
Where did the game of croquet originated?	France
When did Bob Marley die?	1981, May, 1980, 1982

Figure 7: **questions and relative candidates, obtained by Google issuing the answer template of each question as query.**

The author of Web pages can be wrong when he assesses something, so for instance he might believe that Bob Marley died in 1980 (and write so), while he did in 1981. Sorting suggestions by popularity may help in these cases.

Web suggestions are treated as additional keywords to the IR-like query. In case the collection does not "agree" with the Web, this will result in a query that returns no candidate paragraphs. In such cases, we can remove the Web suggestion from the query, and proceed with the default behavior.

In figure 8, we see some examples of the generated queries. In bold face, Web suggestions. As it's easy to notice, for long queries it is likely that the addition of a keyword is irrelevant, because few (if any at all) paragraphs are already returned. However, the addition of the Web suggestion is very effective for short questions, where it can narrow and direct the search. After the first expansion loop, that is if no answer is found, the system is given the chance to "prove the Web wrong":

Run	CWS
pqas21	0.357
pqas22	0.358
pqas23	0.354
resorted	0.438

Figure 9: **results for the TREC 2002 answer set. The last run was not submitted, but has still been obtained from NIST judgments.**

by removing the Web suggestion, we let open the chance of finding a different answer. However, we re-insert the Web suggestion at the last expansion phase, at which, by experience, usually so many paragraphs match, that hardly anything useful is ever returned (only the first 400 paragraphs returned by the query are analyzed, in order to save time). Finally, there is a last "desperate" query that we called *last resort*, in which only the Web suggestion is looked for in the collection.

6 Results

We submitted three runs at this year's TREC (see figure 9). The difference among the three runs laid in different settings for constants and weights that showed little effect.

However, with respect to these three runs, a big mistake was made in sorting the answers. The system is biased toward precision, therefore every NIL answer (there are still about 210 NILs!) should be considered as highly uncertain. Re-sorting the answers so that all the NILs come later (and keeping the relative sorting), still basing on NIST judgments, yielded a considerably better result: a CWS of 0.438.

Interesting results regard the use or not of the Web to narrow the search during the IR phase. Thanks to the doping of the search expression by Web suggestions, the searches for TREC 2002 questions returned about 210.000 paragraphs, of which 1 every 15 in average contained the proper answer string. This is an indication that Web suggestions were, in most cases, correct.

Particularly relevant is figure 10, which is the same as figure 4, but modifying the keyword set by adding the Web suggestions. The figure shows how using Web suggestions we had the chance to answer correctly to 70 more questions than we did without the use of the Web. Unfortunately, not all of the paragraphs with the right answer to these 70 questions made it through all the remaining filters, and the final result (the unofficial 0.438) is not extremely higher than what we'd obtain without using the Web (about 0.40).

	# questions	# correct NILs
0 par. returned	30	13
0 ans. within par.	110	38
1+ ans. within par.	243	//

Figure 10: **same as figure 4, this time using Web suggestions in the query.**

7 Conclusions and further work

2002's TREC featured plenty of systems that used the Web in peculiar ways. PiQASso used it to look for possible answers, in order to find "confirmation" for them in the internal collection. Magnini et al. [7] used it the other way around: to confirm correctness of answers gathered from the internal collection. Yang and Chua [11] used it, more similarly to us, to gather additional keywords to include in the IR-style search.

While we also feature complex semantic answer justification (not unlikely Harabagiu et al. [4]), it turns out our cascade structure of the filters is way too strict, at the point that for more than 200 questions we returned NIL. This is mostly due to the relation matching filter being too "strict" (too syntax-concerned). In that respect, techniques like Harabagiu et al.'s lexical chains seem to be able to lead to significant improvement.

However, as we feel we have improved the search phase, and further improvements will concern, for instance, the deploying of NE-tagging-aware indexing, the quality of the semantic filter is on a completely different dimension, and can therefore be developed independently from the search phase.

Moreover, since Google's excerpts undergo the same filters as the paragraphs found in the internal collection, failures of the relation matching filter affect negatively also the effectiveness of the Web search. That is, very often the answer would actually be within the excerpts returned by Google, but it would be filtered out by either the type or semantic filters. We therefore expect great improvements out of the tuning of the last filter, that can be obtained mainly by better exploitation of WordNet (à

Questions	Queries
Where did the game of croquet originate?	- ((**France**) & (originate) & (game) & (croquet)) - ((game) & (croquet croquets)) - ((game biz) & (croquet croquets)) - ((**France**) & (game biz) & (croquet croquets)) - ((**France**))
Who was the first person to run the mile in less than four minutes?	- ((**Bannister**) & (person) & (run) & (mile) & (minutes) & (less) & (four) & (first)) - ((person) & (run ran run running) & (mile) & (minutes) & (less) & (four) & (first)) - ((**Bannister**) & (mile knot mi) & (minutes) & (first)) - ((**Bannister**))
Who is the Governor of Tennessee?	- ((**Alexander Johnson Menkov Pearsall Ronnie War Hilleary McWhorter**) & (governor) & (Tennessee)) - ((governor governors) & (Tennessee)) - ((**Alexander Johnson Menkov Pearsall Ronnie War Hilleary McWhorter**)

Figure 8: **examples of question/queries pair, at various expansion levels. Words in the inner parenthesis are in or, otherwise in and. The query is a proximity query.**

la lexical chains).

References

[1] G. Attardi et al., *PiQASso: Pisa Question Answering System*, proc. TREC 2001 Conference, Columbia, MD, November 2001

[2] G. Attardi, A. Cisternino, *Reflection support by means of template meta-programming*, proc. of Third International Conference on Generative and Component-based Software Engineering, LNCS, Springer-Verlag, Berlin, 2001

[3] R. Grisham and B. Sundheim, *Design of the MUC-6 evaluation*, proc. of the MUC-6 conference, Columbia, MD, 1995

[4] S. Harabagiu, Moldovan et al., *LCC Tools for Question Answering*, preliminary proc. of TREC 2002 conference, Columbia, MD, November 2002

[5] D. Lin, *LaTaT: Language and Text Analysis Tools*, proc. Human Language Technology Conference, San Diego, California, March 2001

[6] K. C. Litkowsi, *Question Answering Using XML-Tagged Documents* preliminary proc. of TREC 2002 conference, Columbia, MD, November 2002

[7] B. Magnini et al., *Mining Knowledge from Repeated Co-occurrences: DIOGENE at TREC-2002*, preliminary proc. of TREC 2002 conference, Columbia, MD, November 2002

[8] G. Miller, *Five papers on WordNet*, special issue of International Lexicography 3(4), 1990.

[9] J. C. Reyner and A. Ratnaparkhi, *A Maximum Entropy Approach to Identify Sentence Boundaries*, Computational Language, 1997.

[10] G. Schmid, *TreeTagger – a language independent part-of-speech tagger*, 1994. Available: http://www.ims.uni-stuttgart.de/Tools/DecisionTreeTagger.html

[11] H. Yang and T. Chua, *The Integration of Lexical knowledge and External Resources for Question Answering*, preliminary proc. of TREC 2002 conference, Columbia, MD, November 2002

The University of Sheffield TREC 2002 Q&A System

Mark A. Greenwood, Ian Roberts and Robert Gaizauskas
{m.greenwood,i.roberts,r.gaizauskas}@dcs.shef.ac.uk

Department of Computer Science
University of Sheffield
Regent Court, Portobello Road
Sheffield S1 4DP UK

1 Introduction

The system entered by the University of Sheffield in the question answering track of TREC 2002 represents a significant development over the Sheffield system entered into TREC-8 [9] and TREC-9 [15], although the underlying architecture remains the same. The essence of the approach is to pass the question to an information retrieval (IR) system which uses it as a query to do passage retrieval against the text collection. The top ranked passages output from the IR system are then passed to a modified information extraction (IE) system. Syntactic and semantic analysis of these passages, along with the question, is carried out to identify the *"sought entity"* from the question and to score potential matches for this sought entity in each of the retrieved passages. The potential matches are then combined or discarded based on a number of criteria. The highest scoring match is then proposed as the answer to the question.

2 System Description

2.1 Overview

The key features of the question answering system, for processing a single question. are shown in Figure 1. Firstly the TREC document collection is indexed using the probabilistic Okapi information retrieval system (this is done once only in advance of any questions) [14]. This index is then used to return the top n passages relevant to the question. the query to Okapi being the question words. The top n passages are then submitted along with the question to QA-LaSIE, our modified IE system, which should produce one or more answers.

The reasoning behind this architecture is straightforward. The text collection is too large to be processed in its entirety by the IE system. It is. however, the IE system which is capable of carrying out the detailed linguistic analysis needed to answer the questions. IR systems, however, are specifically designed to process huge amounts of text, and to return the result of a query in a short space of time. Using an IR system as a filter between the text collection and the IE system should allow us to benefit from the systems' respective strengths.

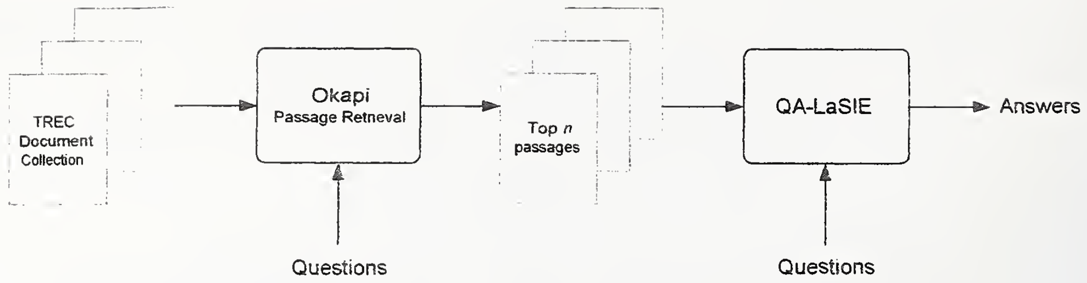


Figure 1: System setup for the question answering task.

Passage Length	*Coverage*	*Correct Answers (out of 100)*
1 paragraph	67%	13
2 paragraphs	74%	11
3 paragraphs	72%	7
4 paragraphs	72%	8
5 paragraphs	70%	7
6 paragraphs	70%	7
7 paragraphs	71%	7
full documents	72%	7

Table 1: Results of IR experiments and their effects on the QA system.

2.2 Okapi

An important question involving the information retrieval component of the QA system is how much text to return. We must decide a) how many documents to retrieve and b) how much retrieved text per document to return (passage size).

For TREC 2002 we decided to process the top 20 documents returned by Okapi. Experimentation not complete at the time of the test run subsequently showed we should have considered about five times this number of documents as on average the top twenty documents only contained answers for about 60% of the questions while the top 100 documents contained on average answers for about 85% of the questions [13] (of course the more documents examined per question, the greater the number of entities which can potentially be confused with the answer).

Okapi supports passage retrieval which can be parameterised by setting a minimum passage length, a maximum passage length and a step value controlling how the passage window is moved over the text. We experimented with different sizes of passage using a random sample of 100 TREC-9 and TREC 2001 questions as queries against the TREC-2001 document collection. These experiments are documented in [13] and their main results are detailed in Table 1. The definition of the data in each column of the table is as follows:

Coverage the percentage of questions for which at least one relevant answer bearing passage was found in the retrieved data.

Correct Answers the number of questions for which the exact answer returned by the system matched one of the Perl patterns supplied for that question.

TREC Score the TREC 2002 confidence score for the run.

From these results it seems that using passages of one paragraph in length gives the best performance (13 of the 100 questions were answered correctly), even though the best coverage is provided for passages of length two paragraphs. As the main experiments were not completed in time, two of the submitted runs used the retrieval techniques previously used with our question answering system (i.e. passages of up to three paragrpahs in length, see [15]) and one run used passages of just one paragraph to attempt to confirm the experimental results.

2.3 LaSIE

The basis of the question answering system is the LaSIE information extraction system. originally developed to participate in the Message Understanding Conference evaluations [8]. LaSIE operates inside the GATE platform [4], and as a new version of GATE has become available [3] since our participation in TREC-9, LaSIE has been ported to use the new version, leading to a few minor changes.

The system is essentially a pipeline of modules each of which process the entire text before the next module is invoked. The following is a brief description of each of the modules in the LaSIE system:

Tokeniser Identifies token boundaries and text section boundaries.

Gazetteer Identifies single and multi-word matches against multiple domain specific full name and keyword lists, and tags matching phrases with appropriate name categories.

Sentence Splitter Identifies sentence boundaries in the text body.

POS Tagger A rule-based part-of-speech tagger [6].

Tagged Morph Simple morphological analysis to identify the root form and inflectional suffix for tokens that have been tagged as noun or verb.

NE Transducer Identifies names of people, organisations etc.

Parser Performs two-pass bottom-up chart parsing, pass one with a special named entity grammar, and pass two with a general phrasal grammar. A best parse is then selected, which may be only a partial parse, and a quasi-logical form (QLF) of each sentence is constructed.

Discourse Interpreter Adds the QLF representation to a semantic net, which encodes the system's world and domain knowledge as a hierarchy of concepts. Additional information inferred from the input is also added to the model, and coreference resolution is attempted between instances mentioned in the text, producing an updated discourse model. A representation of the question is then matched against the model.

2.4 QA-LaSIE

The QA-LaSIE system takes as input a question and a set of passages retrieved by the IR system and outputs the highest ranked answer. When multiple questions are processed by the system it outputs one answer per question ranking the answers based on how confident it is in the answers.

Figure 2 shows the end-to-end layout of the system as entered in TREC 2002. Four key alterations were made to the original LaSIE IE system for entry into the question answering track at TREC-8 and TREC-9 and these have been developed further for this year's entry. These alterations are as follows:

1. the grammar used by the parser was extended to cover question types;

2. the discourse interpreter was modified to allow the QLF representations of each question to be matched against the discourse model of a candidate answer text;

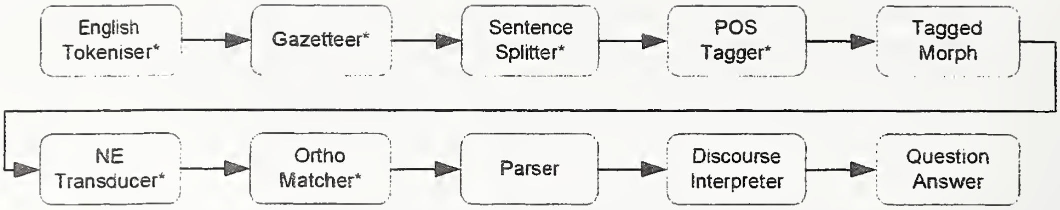


Figure 2: QA-LaSIE system modules (* denotes a standard GATE 2 module).

3. an answer identification procedure which scored all discourse entities in each candidate text as potential answers was added to the discourse interpreter;

4. a Question Answer module was added to examine the discourse entity scores across all passages, determine the ranking of the answers and then output the appropriate answer text.

Exact details of these changes would not sufficiently explain the essence of the approach taken to question answering by the QA-LaSIE system. Therefore the following sections describe the key processes involved in our approach to question answering.

2.4.1 Parsing: Syntactic and Semantic Analysis

Questions were one of the sentence constructions not handled by the original LaSIE parser. Extra grammar rules were developed to cover the example questions that were available. The syntactic grammar rules have a semantic component that is used to build a QLF representation of the question. One major difference between LaSIE and QA-LaSIE is the introduction of a special semantic predicate, `qvar` (question variable), which is used to indicate the entity requested by the question. For example, the question *"Who wrote Hamlet?"* produces the following QLF representation:

```
qvar(e1), qattr(e1,name), person(e1), lsubj(e2,e1),
write(e2), time(e2,past), aspect(e2,simple),
voice(e2,active), lobj(e2,e3), name(e3,'Hamlet')
```

In this representation each entity in the question gives rise to a unique identifier of the form `e`*N*. The use of the word *Who* in the question suggests the answer will be a person and so `person(e1)` is added to the QLF. Also the `qvar` is set to `e1` showing that the question is seeking a person (as `person` and `qvar` share the same entity). The relational predicates `lsubj` (logical subject) and `lobj` (logical object) link any verb arguments found in the text with the verb in the correct relationship.

The QLF representation of the question is stored for use in subsequent processing against the candidate answer texts and the entity identifiers are replaced by question entity identifiers of the form `q`*N* (i.e. `e1` becomes `q1`, `e2` becomes `q2` etc.) to facilitate later processing.

Candidate answer texts are processed in exactly the same fashion although the grammar rules do not instantiate a `qvar` and the entity identifiers are not altered.

2.4.2 Resolution of Question and Candidate Answer Texts

After a candidate answer text has been parsed the QLFs are passed to the discourse interpreter. This behaves as in the LaSIE system apart from the addition of a final processing stage.

The discourse interpreter has (by this stage) produced a semantic net or discourse model of all the entities and relationships present in the multiple QLFs for a document. This is built by running a coreference algorithm against the semantic representation of successive sentences as they become available, in order to unify them with the discourse model built so far. This results in multiple references to the same entity across the text being merged into a single unified instance.

Given this discourse model of a text, the QLF of the question is added to the model as the first sentence and coreference is then carried out between question entities (qN) and entities within the text (eN).

The method for determining and scoring each candidate answer is then as follows:

1. Each sentence in a candidate answer document is given a constraint score, C, equal to 1 point for each question constraint that matches a member of the sentence, where a question constraint is a unary predicate specifying the type of an entity (eg. person is the type of `person(eY)` in the question.

2. Within each sentence every remaining entity (eY) is tested for:

   (a) Semantic Similarity to the `qvar`, S: the reciprocal of the length of the path between the type of the `qvar` entity and the type of `eY` in the semantic lattice (ontology) or if this fails (usually because the two entities are not both present in the system's small ontology) the reciprocal of the Leacock-Chodorow distance [10] between the `qvar` and `eY` in WordNet [12]. For instance if `qvar` and `eY` are of the same type then they will receive a score of 1.

   (b) Object Relation, O: 0.25 if `eY` is related to a question constraint within the sentence by apposition, a qualifying relationship, or with the prepositions *of* or *in*.

   (c) Event Relation, E: 0.5 if there is an event entity in the QLF of the question which is related to the `qvar` by a `lsubj` or `lobj` relation and is not the `be` event and `eY` stands in the same relation to an event entity of the same type as `qvar` does.

These three values are then combined with the scores for the sentence and the number of question constraints, Q, to give Equation 1 (where 2.8 is a normalising factor determined via experimentation).

$$\text{Score for eY} = \frac{(\frac{S+O+E}{2.8}) + C}{1 + Q} \tag{1}$$

The discourse interpreter then returns all the candidate answers and their associated scores for processing by the answer module.

2.4.3 Answer Output and Ranking

Due partly to the differences between TREC 2002 and the previous question answering tracks and partly to the move to using the new version of GATE, the final Question Answering module has been completely redeveloped and a number of new ideas have been included, which are outlined in this section.

The limitations of window-based methods for pinpointing answers have been discussed in numerous papers including [7]. The main concerns with these methods are:

- It is impossible to accurately pinpoint the boundaries of an answer (e.g. an exact name or phrase).

- These rely solely on word level information and do not use semantic information (hence no knowledge of the type, such as person or location, of the answer being sought).

- It is impossible to see how such methods could be extended to composing an answer from many different documents or even from different sentences or phrases within a single document.

One way to filter out some inappropriate candidate answers is to assume that overlap between the question and a candidate answer is inherently bad. Clearly for a question such as *"Where is Perth?"* an answer of *"Perth is in"* is not correct and can be eliminated using the following method.

In most cases it is unlikely that a correct exact answer to a question will contain many, if any, of the non-stopwords in the question. We can use this assumption to throw away some of the candidate answer strings before we even look at the score assigned to them. Word overlap between a question and candidate answer can be expressed as a percentage. At 0% there is no overlap between the question and candidate answer and so the string may be a correct answer to the question and therefore requires further processing. At 100% overlap all the non-stopwords in the candidate answer appear in the question, at which point it is highly unlikely that this string will be a correct answer to the question and can therefore be discarded (an exception is TREC 2001 question 1026 *"What does target heart rate mean?"* which has as one of its possible answers *"target heart rate"*, although the more important question here is whether *"target heart rate"* is in fact a valid answer to the question). At points between 0% and 100% overlap it is unclear whether the candidate answer may or may not be correct. Our system simply discards any candidate answers which overlap 100% with the question they seek to answer.

Having carried out some limited analysis of the performance of our system over the TREC 2001 questions, one thing was clear; we would often return two or more semantically equivalent answers. Clearly if the answer is correct then this is alright, but if these answers are wrong then this may well prevent correct answers from appearing in the top n answers which we are allowed to return. On some occasions we were actually returning identical answers (i.e. for Q1000 *"The sun's core, what is the temperature?"* we returned five answers all of which were *"the sun"*), these are easy to remove by simply keeping only the highest scoring of two identical answers.

Furthermore it may be possible to prune candidate answers that are substrings of longer candidate answers, when the question is suitably vague; as is the case in the question *"Where is Perth?"* to which our system returns a list of ranked answers containing: *Australia* and *Western Australia*. Clearly *Australia* and *Western Australia* are both acceptable answers to the question, so only one of them need be returned.

The approach taken to deal with these answer strings, similar to that used in [1], is to test if two proposed answers A and B are the similar by checking that the stem of every non-stopword in A matches a stem of a non-stopword in B, or vice versa. Using this test, if two answers match, then both are removed and a new answer is created from the highest of the two scores and the longest answer string. The effect of this method on our example question was that now only *Western Australia* is listed as a possible answer.

Applying the same approach to the question *"In which country is Perth?"* would not be as effective, since *Western Australia* is not an exact country name so while this method is better than simple string matching approaches, there is still scope for improvement.

Using this approach improved the system performance slightly. More importantly was the unexpected side effect which caused the system to clarify some answer strings, with the most obvious being peoples names: *'Armstrong'* becomes *'Neil A. Armstrong'* and *'Davis'* becomes *'Eric Davis'*, etc.

These techniques (overlap, similar answers, etc.) are then used to discard, merge and rank the candidate answers found within the document collection for a single question (full details of the ranking algorithm can be found in [5]).

The method of ranking single answers to multiple questions (i.e. to produce the confidence sorted list required for this years submission) is based on the following attributes of each answer:

- the score (the higher the better)
- the number of other answers which were semantically the same as this one (the higher the better)

- the IR system rank of the document from which the answer originates (the lower the better as the top document returned by the IR step is ranked 1)

2.4.4 Answering Questions Requiring Multiple Answers

List questions are inherently harder to answer than standard, single answer questions, mainly because systems have to combine information from multiple sources to locate the required number of answers. Also a system has to be able to extract from the question the number of different answers required.

Our simple solution to these problems is as follows:

1. The system processes the question in the usual way, producing a long list of ranked answers.
2. The question is then scanned, token by token, until the first token whose part-of-speech signifies that it is a number. This is then assumed to be the number of answers sought.
3. The requested number of answers is then returned from the top of the ranked list.

Clearly this approach suffers from the problem that some questions may contain more than one number, i.e. *"In the 2001 US Presidential election who were the 2 main candidates?"*. This problem did not surface during the evaluation, however, as our system correctly identified the number of answers to return for all the questions.

2.4.5 Boosting Performance using Answer Redundancy

As has been reported in Light et al. [11], the number of answer instances to a given question in the document collection (within a single document or multiple documents each containing the answer once) directly effects the end-to-end performance of a QA system. This is partly due to the fact that the IR engine is more likely to find a relevant document, and also because the answers are likely to occur in different contexts, giving the parser a better chance of analysing at least one of them in a way that is beneficial to the rest of the system.

To this end it was decided to attempt to boost the knowledge available to our system, not as may be expected, by returning more documents at the initial IR step, but by using two different text collections. The second text collection that was chosen was the World Wide Web. A document collection for a single question is constructed from the snippets displayed on the Google results page for the top ten documents returned by Google. These snippets are certainly not full documents, and are rarely full sentences but this is not a problem as the bottom-up chart parser we employ is not constrained to only selecting full sentence or complex phrase categories. This method of using just the snippets has been shown to be successful in [2], although they used the snippets from the first one thousand documents rather than the first ten.

The QA system is run against both text collections and then the results are merged together. The end result must be an answer which references a document in the TREC collection so the process of merging is as follows: for each answer returned from the Google corpus, if an answer exists from a document in the TREC corpus which is semantically equivalent, then merge by keeping the highest score etc., but the reference to the TREC document (other answers found using Google are simply discarded).

Over a sample of one hundred questions (TREC questions 1000 to 1099) the results of combining the collections in this way (based on returning the top five answers for each question) can be seen in Table 2.

Collection	*MRR*	*Not Found (%)*
TREC	0.256	68 (68%)
Google	0.227	68 (68%)
Combined	0.285	65 (65%)

Table 2: Results of using Google to boost system score.

		Not			*Confidence*	*No Answer*	
Run Tag	*Wrong*	*Supported*	*Inexact*	*Right*	*Score*	*Precision*	*Recall*
`sheft11mo3`	422	9	18	51	0.128	0.162	0.130
`sheft11mog3`	394	12	22	72	0.203	0.150	0.130
`sheft11mog1`	389	11	20	80	0.222	0.150	0.065

Table 3: Results from the three main track entries.

3 Results and Analysis

3.1 Results Observed During Development

Unfortunately we did not enter the system into TREC 2001 and so there were no official scores for the system over that question set. However, the first development task was to produce unofficial scores for our unaltered TREC-9 system over the TREC 2001 questions, using the regular expression patterns kindly made available by NIST. The result was that the unaltered TREC-9 system achieved a mean reciprocal rank (MRR) score of 0.169, over the TREC 2001 questions compared to its official TREC 9 MRR score of 0.206 (both using answers of 50 bytes or less). This drop in performance may be due to the fact that our system is not designed to handle definition style questions, which made up a significant percentage of the TREC 2001 question set. This system now scores an MRR of 0.343 over the TREC 2001 question set, clearly a significant improvement over the previous system.

3.2 Final Evaluation Results

3.2.1 Main Track

We submitted three runs to the main question answering track. The differences between the runs all concernes the size and composition of the document collection generated for a single question, these were:

`sheft11mo3`: This run used the top twenty passages retrieved from the AQUAINT collection by Okapi. The maximum length of a passage was three paragraphs, the minimum was one paragraph.

`sheft11mog3`: This run used the same collection as did `sheft11mo3`, augmented with the top ten snippets returned by Google when given the question as a search query.

`sheft11mog1`: This run is the same as `sheft11mog3` except the passages retrieved by Okapi from the AQUAINT collection are limited to at most one paragraph in length.

Table 3 shows the full evaluation results for the three different runs over the 500 test questions. From this it can be seen that the sheft11mog1 run was the best of the three configurations, suggesting that using documents from more than one source is beneficial, and also that documents of one pargraph in length are more suited to this work than longer documents, confirming what was demonstrated during development (see Section 2.2).

3.2.2 List Track

We submitted two runs to the list track. The differences between the runs concern the size and composition of the document collection generated for a single question, these were:

`sheft11lo`: This run used the top twenty passages retrieved from the AQUAINT collection by Okapi. The maximum length of a passage was a single paragraph.

`sheft11log`: This run used the same collection as `sheft11lo`, augmented with the top ten snippets returned by Google when given with the question as a search query.

Unfortunately we did not have time to test the list answering system with the result that the system scored an average accuracy of only 0.06 (for both runs). A serious flaw in the processing of list questions was subsequently discovered, although fixing this resulted in a system whose average accuracy was only 0.09.

4 Conclusions and Future Work

At its core, Sheffield's entry in this year's QA track remains the same as our TREC-9 system in 2000 [15]. There were, however, a number of enhancements, the most significant were:

- using a semantic similarity metric over WordNet as one factor in determining the score of candidate answer entities;
- filtering the final ranked answer list
  - to remove duplicate and near-duplicate answers and simultaneously boost the remaining candidate's rank;
  - to eliminate answers which completely overlap with the question;
- employing Google to search the Web for documents relevant to a given question and boosting the rank of answers found by the QA system both in the Google-returned document snippets and the TREC collection.

Each of these enhancements produced small but noticeable improvements. Ideas for future work include:

- expanding the size of the document set passed on from the IR system to the QA system – experiments not completed till after the TREC run showed that for 40% of the questions in a test sample the QA system was simply not receiving any document containing an answer;
- experimenting with adaptive algorithms to optimise the weightings of the various features used to rank the answer candidates.

References

[1] Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. Data-Intensive Question Answering. In *Proceedings of the Tenth Text REtrieval Conference*. 2001.

[2] S. Bucholz and W. Daelemans. Complex Answers: A Case Study using a WWW Question Answering System. *Natural Language Engineering*, 7(4), 2001.

[3] H. Cunningham. GATE, a General Architecture for Text Engineering. *Computers and the Humanities*, 36:223–254, 2002.

[4] R. Gaizauskas, H. Cunningham, Y. Wilks, P. Rodgers, and K. Humphreys. GATE – an Environment to Support Research and Development in Natural Language Engineering. In *Proceedings of the 8th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-96)*, pages 58–66, Toulouse, France, October 1996.

[5] Mark A. Greenwood. Question Answering. First year PhD Progress Report, Department of Computer Science, The University of Sheffield, UK. Available, October 2002, from http://www.dcs.shef.ac.uk/~mark/phd/work/index.html, 2002.

[6] Mark Hepple. Independence and Commitment: Assumptions for Rapid Training and Execution of Rule-based POS Taggers. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-2000)*, pages 278–285, Hong Kong, October 2000.

[7] Eduard Hovy, Laurie Geber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. Towards Semantics-Based Answer Pinpointing. In *Proceedings of the DARPA Human Language Technology Conference (HLT)*, San Diego, CA, 2001.

[8] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, and Y. Wilks. Description of the LaSIE-II system as used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998.

[9] Kevin Humphreys, Robert Gaizauskas, Mark Hepple, and Mark Sanderson. University of Sheffield TREC-8 Q & A System. In *Proceedings of the 8th Text REtrieval Conference*, 1999.

[10] C. Leacock and M. Chodorow. Combining Local Context and WordNet Similarity for Word Sense Identification. In C. Fellbaum, editor, *WordNet: An Electronic Lexical Database*, chapter 11, pages 265–285. MIT Press, 1998.

[11] Marc Light, Gideon S. Mann, Ellen Riloff, and Eric Breck. Analysis for Elucidating Current Question Answering Technology. *Natural Language Engineering*, 7(4), 2001.

[12] George A. Miller. WordNet: A Lexical Database. *Communications of the ACM*, 38(11):39–41, November 1995.

[13] Ian Roberts. Information Retrieval for Question Answering. MSc Dissertation, Department of Computer Science, The University of Sheffield, UK. Available, Februray 2003, from http://www.dcs.shef.ac.uk/teaching/eproj/msc2002/abs/m1ir.htm, 2002.

[14] S. Robertson and S. Walker. Okapi/Keenbow at TREC-8. In *Proceedings of the 8th Text REtrieval Conference*, 1999.

[15] Sam Scott and Robert Gaizauskas. University of Sheffield TREC-9 Q & A System. In *Proceedings of the 9th Text REtrieval Conference*, 2000.

Natural Language Based Reformulation Resource and Web Exploitation for Question Answering

Ulf Hermjakob, Abdessamad Echihabi, Daniel Marcu

Information Sciences Institute
University of Southern California
{ulf, echihabi,marcu}@isi.edu

Abstract

We describe and evaluate how a generalized natural language based reformulation resource in our TextMap question answering system improves web exploitation and answer pinpointing. The reformulation resource, which can be viewed as a clausal extension of WordNet, supports high-precision syntactic and semantic reformulations of questions and other sentences, as well as inferencing and answer generation. The paper shows in some detail how these reformulations can be used to overcome challenges and benefit from the advantages of using the Web.

1 Introduction

Question answering can be easy and it can be very hard. The degree of difficulty doesn't primarily depend on the question *per se*, but rather on how closely a given corpus matches the question.

Q: Who discovered America?
S1: Columbus discovered America.
S2: Columbus Day celebrates the Italian navigator who first landed in the New World on Oct. 12, 1492.

The question above can be answered more easily from sentence S1 than sentence S2 because the string Q is "closer" to string S1 than string S2. Brill et al. (2001) have already demonstrated that a good methodology for increasing the performance of a QA system is that of using the Web as an additional textual resource. When a QA system looks for answers within a relatively small textual collection, the chance of finding strings/sentences that closely match the question string is small. However, when a QA system looks for strings/sentences that closely match the question string on the web, the chance of finding correct answers is much higher. And once a QA system knows the appropriate answer, it is much easier to find support for it in the original collection.

The approach proposed by Brill et al. reduces the gap between questions and answers by searching a larger pool of textual material. In this paper, we propose an alternative approach to reducing the gap between questions and potential answers. In our approach, we create semantically equivalent paraphrases of the question given as input and we assume that a correct answer is found whenever an answer sentence/string matches any of the paraphrases that are equivalent to the question given as input.

For example, the question "How did Mahatma Ghandi die?" is automatically paraphrased by our TextMap question answering system into 30 variants, some of which are shown below, along with paraphrases of two other questions.

1446.0: How did Mahatma Gandhi die?
1446.1: Mahatma Gandhi died <how>
1446.2: Mahatma Gandhi died of <what>
1446.3: Mahatma Gandhi died from <what>
1446.4: Mahatma Gandhi's death from <what>
1446.5: Mahatma Gandhi drowned
1446.6: Mahatma Gandhi suffocated
1446.7: Mahatma Gandhi froze to death
1446.8: <who> killed Mahatma Gandhi
1446.9: <who> assassinated Mahatma Gandhi
... ...
1446.30: Mahatma Gandhi was killed

1439.0: How deep is Crater Lake?
1439.1: Crater Lake is <what distance> deep
1439.2: depth of Crater Lake is <what distance>
1439.3: Crater Lake has a depth of <what distance>
1439.4: <what distance> deep Crater Lake

1772.0: Who invented the cotton gin?
1772.1: <who> invented the cotton gin
1772.2: <who> was the inventor of the cotton gin
1772.3: <who>'s invention of the cotton gin
1772.4: <who> was the father of the cotton gin
1772.5: <who> received a patent for the cotton gin

As one can see, the paraphrases cover a fairly wide range of reformulations, from syntactic, such as reformulations 1446.1 and 1439.1, to semantic, such as reformulations 1439.2 and 1772.2. Some reformulations, such as 1446.8 and 1772.5 can be even interpreted as a rudimentary example of natural language inferences.

Our results show that the ability to paraphrase questions is useful with respect to two question answering subprocesses.

- First, question paraphrases can be used in conjunction with a retrieval engine in order to find documents that are more likely to contain correct answers than documents that are retrieved using standard query formulation techniques.
- Second, question paraphrases can be used in order to rank and select better answers than those that are selected by a system that does not have paraphrasing capabilities.

In this paper, we first describe briefly (Section 2) the techniques we use in order to generate paraphrases/reformulations. In Section 3, we present experiments that underscore the utility of question paraphrases in the context of retrieving documents that contain correct answers to questions. In Section 4, we evaluate the utility of our paraphrasing capability in the context of a complete question answering system.

2 Reformulations

To address the problem of mismatches going beyond relatively simple variations such as synonyms and alternative spellings, we are building a collection of phrasal synonyms in extended natural language format. Here are some examples of our reformulations:

```
:anchor-pattern "SOMEBODY_1 is the spouse of SOMEBODY_2." :reflexive t
:is-equivalent-to "SOMEBODY_1 is married to SOMEBODY_2." :reflexive t
:can-be-inferred-from "wedding of SOMEBODY_1 and SOMEBODY_2." :reflexive t
:can-be-inferred-from "SOMEBODY_1 married SOMEBODY_2." :reflexive t
:rebutted-by "SOMEBODY_1 and SOMEBODY_2 divorced." :reflexive t

:anchor-pattern "SOMETHING_1 costs MONETARY_QUANTITY_2."
:is-equivalent-to "the price of SOMETHING_1 is MONETARY_QUANTITY_2."
:is-equivalent-to "SOMETHING_1 is on sale for MONETARY_QUANTITY_2."
:can-be-inferred-from "to buy SOMETHING_1 for MONETARY_QUANTITY_2."

:anchor-pattern "SOMEBODY_1 sells SOMETHING_3 to SOMEBODY_2."
:is-equivalent-to "SOMEBODY_2 buys SOMETHING_3 from SOMEBODY_1."

:anchor-pattern "SOMEBODY_1 died of SOMETHING_2."
:is-equivalent-to "SOMEBODY_1 died from SOMETHING_2."
:is-equivalent-to "SOMEBODY_1's death from SOMETHING_2."
:answers "How did SOMEBODY_1 die?" :answer SOMETHING_2

:anchor-pattern "SOMEBODY_1 died from a specific cause." :intermediate-only
:can-be-inferred-from "SOMEBODY_1 drowned."
:can-be-inferred-from "SOMEBODY_1 suffocated."
:can-be-inferred-from "SOMEBODY_1 froze to death."
:answers "How did SOMEBODY_1 die?" :answer :full-pattern

:anchor-pattern "SOMEBODY_2 killed SOMEBODY_1."
:can-be-inferred-from "SOMEBODY_2 assassinated SOMEBODY_1."
:answers "How did SOMEBODY_1 die?" :answer :full-pattern :passive-answer
```

Our system first parses a given question and then identifies its answer type (Hovy 2001). The question reformulation module then uses parsed versions of reformulation patterns as shown above to produce high-precision meaning-preserving variants of the question. The objective of these reformulations is to increase the likelihood of finding correct answers in texts.

The number of reformulations produced by our current system varies from one reformulation (which might just rephrase a question into a declarative format) to about 30 reformulations, with an average of currently 3.14. The reformulation collection currently contains 420 assertions grouped into about 100 equivalence blocks. Many of them are manual generalizations of automatically derived patterns generated by (Ravichandran, 2002).

The principal advantage of using an extended natural language to express phrasal synonyms is that the format is very intuitive for humans. Phrasal synonyms are easy to write, or, when they are automatically generated, easily checked and filtered.

The expressiveness and focus of the patterns is greatly enhanced by variables that can carry syntactic or semantic restrictions. Compared to automatically generated patterns such as (Ravichandran, 2002) and (Lin, 2001), there is also no limit on the number of variables per reformulation and, since all patterns have been checked by hand, only few misreformulations.

In order to fully exploit the power of phrasal synonyms, our TextMap system parses all reformulation patterns and then performs the actual reformulation and matching at the parse tree level. This allows sentences and patterns to match even if the word order differs, as is often the case when sentences differ in mode (interrogative/declarative), voice (active/passive), or the presence of intervening constituents such as relative clauses or prepositional phrases.

Reformulations are useful even when they don't match any sentence at the surface level. Example:

Question: How deep is Crate Lake?
Reformulation: Crater Lake has a depth of <what distance>
Text: Crater Lake, with a depth of 1,932 feet, ...
Answer: 1,932 feet

While the text might not contain the exact phrasing of the reformulation, it does provide 'depth' as a valuable search term for the document retrieval engine and also allows the matcher to identify the correct answer with a higher confidence.

Our TextMap system uses the collection for the reformulation of questions, but the phrasal synonyms are not inherently question-oriented. The assertion

```
:anchor-pattern "SOMEBODY_1 sells SOMETHING_3 to SOMEBODY_2."
:is-equivalent-to "SOMEBODY_2 buys SOMETHING_3 from SOMEBODY_1."
```

for example also allows the reformulation from one declarative sentence to another:

- John sold the laptop to Mary.
- Mary bought the laptop from John.

By expressing reformulations in natural language, for both the reformulation patterns in the internal reformulation file and the reformulation output for a given question, the reformulation module is kept highly independent of other parts of a QA system, making it easily reusable for other QA architectures and other natural language applications.

2.1 Syntactic Reformulations

Even when reformulations are only fairly shallow (syntactic), the benefits turn out to be significant. All the reformulation mechanism might do is turn a question in interrogative word order such as "When did the Titanic sink?" into the corresponding declarative form: "The Titanic sank <when>". As described in much more detail in the next section of this paper, this can be greatly exploited with search engines that allow multi-word search strings.

Here are some more examples of relatively simple reformulations:

1439.0:	How deep is Crater Lake?
1439.1:	Crater Lake is <what distance> deep
1439.2:	<what distance> deep Crater Lake
1439.3:	Crater Lake has a depth of <what distance>
1439.4:	depth of Crater Lake is <what distance>c
1782.0:	Who was the first woman to run for president?
1782.1:	<who> was the first woman to run for president
1782.2:	the first woman to run for president was <who>
1782.3:	<who>, the first woman to run for president

2.2 Advanced forms of reformulation

Reformulation patterns are general enough to support inferencing, reformulation chains, and generation.

2.2.1 Inference

While some assertions are truly equivalent, reformulation is often only a one-way street:

```
:anchor-pattern "SOMEBODY_1 invented SOMETHING_2."
:can-be-inferred-from "SOMEBODY_1 received a patent for SOMETHING_2." :weight 0.8
```

Question: Who invented the telephone?
Reformulation: <who> received a patent for the telephone
Text: Alexander Graham Bell received a patent for the telephone.
Answer: Alexander Graham Bell

2.2.2 Reformulation Chains

Reformulations can be chained together, as the following example shows:

```
:anchor-pattern "SOMEBODY_1 is a student at COLLEGE_2."
:answers "Where does SOMEBODY_1 go to college?" :answer COLLEGE_2

:anchor-pattern "SOMEBODY_1 was a student at COLLEGE_2."
:can-be-inferred-from "SOMEBODY_1 dropped out of COLLEGE_2."

:anchor-pattern "SOMEBODY_1 dropped out of COLLEGE_2."
:is-equivalent-to "SOMEBODY_1 is a COLLEGE_2 dropout."
```

Question: Where did Bill Gates go to college?
Reformulation: Bill Gates was a student at <which college>
Reformulation: Bill Gates dropped out of <which college>
Reformulation: Bill Gates is a <which college> dropout.
Text: Bill Gates is a Harvard dropout.
Answer: Harvard

2.2.3 Generation

Reformulations can also be used to generate answers that are more natural than the actual words in the underlying text:

```
:anchor-pattern "PERSON_1 invented SOMETHING_2."
:is-equivalent-to "PERSON_1 was the inventor of SOMETHING_2."
:is-equivalent-to "PERSON_1's invention of SOMETHING_2"
:answers "Who is PERSON_1 ?" :answer "the inventor of SOMETHING_2"
```

Question: Who was Johan Vaaler?
Reformulation: Johan Vaaler's invention of <what>
Text: ... Johan Vaaler's invention of the paper clip ...
Answer: the inventor of the paper clip

Question: Who was Johan Vaaler?
Reformulation: Johan Vaaler invented <what>
Text: ... the paper clip, invented by Johan Vaaler, ...
Answer: the inventor of the paper clip

2.3 Reformulation Patterns as a Qualitative Extension of WordNet

WordNet(Miller/Fellbaum, 1998) has proven to be a valuable resource for question answering and other natural language applications. Reformulation can be viewed as a qualitative extension of WordNet:

Word synonym	Phrasal synonym
astronaut	SOMETHING_1 is DISTANCE_QUANTITY_2 long.
cosmonaut	length of SOMETHING_1 is DISTANCE_QUANTITY_2.

Generalization	Phrasal inference
lawyer	SOMETHING_1 is the capital of LOCATION_2.
professional person	SOMETHING_1 is in LOCATION_2.

3 Information Retrieval and the Web

We have extended our system to make use of both the TREC collection and the Web. When querying the TREC collection, we use the IR system developed for Webclopedia (Hovy et al., 2001), which uses MG (Witten et al., 1994) as a search engine. When querying the Web we use a new Web-based IR system developed specifically to overcome the challenges and benefit from the advantages of using the Web. The main components of the Web-based IR consist of a Query Reformulation module, a Web search-engine, and a Sentence Ranking module.

3.1 Query Reformulation

One of the main challenges of using the Web as a resource for QA is the generation of Web search-engine queries from a natural language question. A couple of previous systems addressed this challenge. Simple but exhaustive string-based manipulations were used by Brill et al. (2001), Transformational Grammar was used by Kwok et al. (2001) and even learning algorithms were developed by Agichtein et al. (2001) and Radev et al. (2001) to find the best query reformulations.

Our approach was first to analyze how people will naturally form queries to find the answer to a given question. For this purpose, we carried out the following experiment: We randomly selected 50 TREC8 questions and, for each question, we manually produced the simplest queries that yielded the most Web pages containing the answer.

We analyzed the manually produced queries and identified seven "natural" techniques that we used to go from a natural language question to a Web search engine query. We enumerate these techniques below and provide for each an example of natural language question and corresponding Web query.

1. We quoted some question terms and we preserved already quoted terms:
   Name a film that has won the Golden Bear in the Berlin Film Festival?
   Film AND won AND "Golden Bear" AND "Berlin Film Festival"
2. We introduced the appropriate unit if the answer is a quantity:
   How tall is Mt. Everest?
   "Mt. Everest" AND feet
3. We quoted and used parts of the declarative form of the question:
   What is the name of the highest mountain in Africa?
   "the highest mountain in Africa is"
4. We reformulated the declarative form of the question, quoted and used parts of it:
   Where is the Taj Mahal?
   "the Taj Mahal is located"
5. We expanded quoted terms with appropriate prepositions:
   When did Jaco Pastorius die?
   "Jaco Pastorius died on" OR "Jaco Pastorius died in"
6. We used different spellings of some question terms:
   Who was the 16th President of the United States?
   "was the 16th President of the United States" OR "was the 16th President of the U.S."
7. We restricted the Web search to specific Web sites based on the question type:
   Who was the lead actress in the movie "Sleepless in Seattle"?
   site:www.imdb.com actress AND "Sleepless in Seattle"

We, then, derived algorithms that replicate each of the first six observed techniques. We left the last technique for future work. We have also implemented the standard query expansion by morphological variants and synonyms. Table 1 summarizes the different query reformulation techniques used by our Web-IR.

3.2 Sentence Ranking

Using all the query reformulation techniques described earlier, we produce, for each question, a list of Boolean queries (an average of 8 queries per question on TREC-2002). Then, using a Web search-engine, we retrieve the top 10 results (URLs + snippets) for each query. After filtering duplicate URLs, we retrieve the documents, strip HTML, and segment text into sentences.

Every sentence is then scored according to two different schemas:

Name	*Description*	*Sample System Output*
Simple	Preserve quoted terms and quote the smallest NPs.	**What is the longest river in the United States?** -"longest river" AND "United States"
Units	Expand the query with potential units in answer based on answer type and question context.	**How tall is Mt. Everest?** -"Mt. Everest" AND "tall" AND ("foot" OR "feet" OR "miles")
Morpho	Expand the query using morphological variants produced by the Contex parser.	**When did the Titanic sink?** -"Titanic" AND ("sink" OR "sank" OR "did sink")
Synonym	Expand the query using WordNet synonyms.	**What is the length of border between Ukraine and Russia?** -("length" OR "distance") AND ("border" OR "surround") AND ("Ukraine" OR "Ukrayina") AND ("Russia" OR "Soviet Union") AND ("between" OR "betwixt")
Spelling	Expand the query using different spellings of question terms.	**Where is Al-Qaeda located?** -("Al-Qaeda" OR "Al Qaida") AND "located"
Rephrase	Using Contex reformulations, add quoted rephrases of the question's declarative form.	**What is an atom?** -"is an atom" -"an atom is" -"an atom is one of" -"an atom is defined to be" -"an atom is defined as" -"such as an atom" -"called an atom"
Cuephrase	Expand the rephrases with prepositions and/or discourse cues based on answer type.	**When did Abraham Lincoln die?** -"Abraham Lincoln died" -"Abraham Lincoln died on" -"Abraham Lincoln died in"

Table 1: Query Reformulation Techniques

- Scoring with respect to the question/queries terms:
  - Each word in a query is assigned a weight using an information content measure.
  - Each quoted term in a query is given a weight, which is the sum of the weight of its words.
  - Sentences are then scored according to their weighted overlap with the question/queries terms.
- Scoring with respect to the answer:
  - Sentences are tagged using the BBN's IdentiFinder (Bikel et al. 1999).
  - Sentences are then scored according to their overlap with the answer type. The overlap is measured by checking the answer type against IdentiFinder semantic entities in the sentence candidates.

3.3 Web-IR Evaluation:

We have evaluated our Web-IR on the first 200 questions from TREC-2002. We started with the simplest IR system (Simple). And then we added to the system increasingly sophisticated query expansion techniques. The system label Morpho, for example, uses the query expansion techniques that were labeled above as Simple, Unit, and Morpho. Figure 1 shows the percentage of questions that can be answered correctly if one manually selects a sentence from the top 300 or top 10 sentences as ranked by our retrieval component. The results in figure 1 show that more sophisticated query reformulation techniques lead to better candidates for the answer pinpointing module. The most significant improvement occurs when the query rephrasing is added to the pool of query expansion techniques. The clear difference between the percentage of answers found within the top 300 sentences and within the top 10 sentences indicates that further improvement of the sentence ranking module is required.

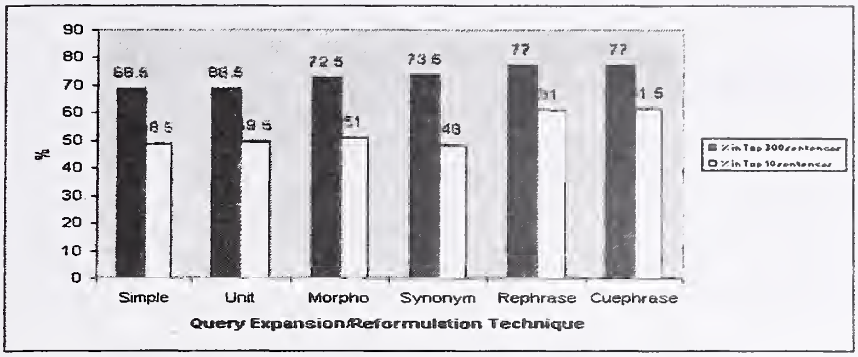


Figure 1: Percentage of answers found in top 300 and top 10 sentences using AltaVista with different query reformulation techniques.

Figure 2 shows how the query reformulation techniques affect the answer redundancy. Intuitively, the more redundant an answer is, the more likely a question answering system is able to find it. Once again, the significant improvement is recorded when the rephrase technique was introduced.

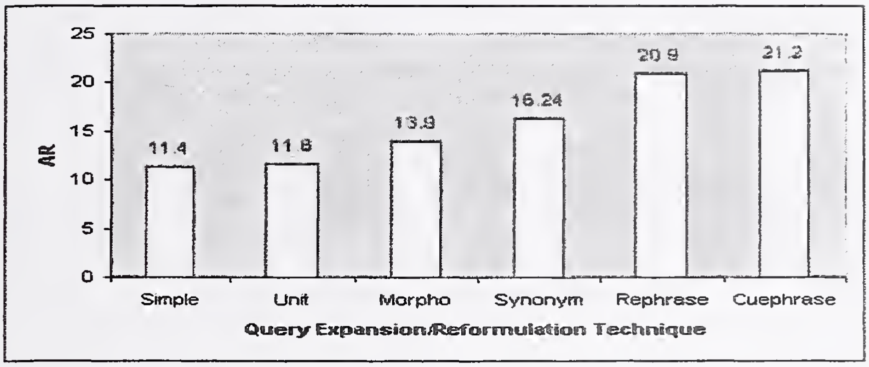


Figure 2: Average Answer Redundancy using AltaVista with different query reformulation techniques.

Finally, figure 3 shows the impact of the search engine choice on the performance of our Web-IR.

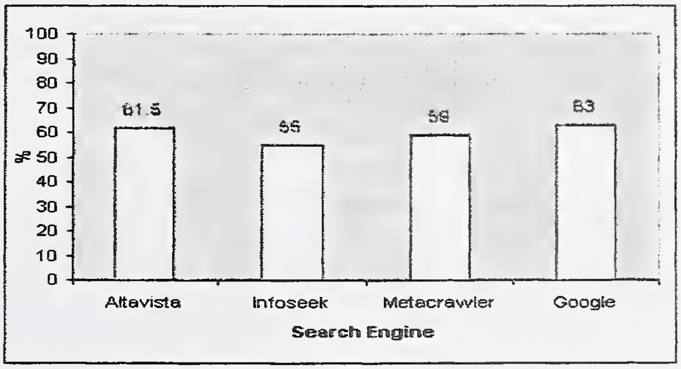


Figure 3: Percentage of answers found in top 10 sentences using different search engines.

4 Evaluation of overall system

We also evaluated the impact of reformulation on overall performance of our system[1]. In the six configurations in table 2, we tested our system on (1) the TREC-2002 collection only, (2) the Web only and (3) the TREC-2002 collection with feedback from the Web, both with (+) and without (-) reformulations.

The results in Table 2 show that our reformulations did not have a significant impact when used in conjunction with a small corpus, the TREC-2002 corpus. The increase in performance for this corpus was small, from 144 to 146 correct answers. However, when used in conjunction with a large corpus, the Web, reformulations enabled us to find many more correct answers (227 compared to 173). Retrofitting the answer to the TREC-2002 collection reduces the performance of our system, but the impact of our reformulation capability remains strong (176 vs. 147) correct answers. These results were achieved with a version of the reformulation module that produced, on average, 1.2 reformulations per question.

Collection	reformulations	R	R+X	R+U	R+X+U	W
TREC only	-	**144**	155	148	160	340
Web only	-	**173**	198	173	198	302
Web+TREC	-	**147**	160	163	178	322
TREC only	+	**146**	157	151	163	337
Web only	+	**227**	251	227	251	249
Web+TREC	+	**176**	187	195	208	292

Note: R stands for the number of questions with a fully correct answer, X for correct but inexact, U for correct but unsupported, and W for wrong.

Table 2: Overall System Evaluations (July 2002 system)

Since July, we have expanded on our reformulation module significantly, motivated by questions and answers from TREC-9. Currently, we are averaging about 3.14 reformulation per question. When we re-evaluated the performance of our system on the first 100 questions in the TREC-2002 collection, we observed that this increase in reformulation capability led to a 5% increase in overall system performance (see Table 3 below).

Collection	reformulations	#ref/q	R	R+X	R+U	R+X+U	W
Web+TREC	+ (July)	1.2	**34**	38	38	42	58
Web+TREC	+ (Oct.)	3.1	**39**	44	41	46	54

Table 3: Effect of additional reformulations

5 Conclusion

Question reformulations increase the likelihood that correct answers are properly identified by (1) enabling the information retrieval module to produce higher quality answer candidates, particularly if the underlying search engine can handle multi-word strings. (2) Reformulations also increase the scoring precision for answer candidates and improve answer pinpointing in the QA matching phase. (3) Finally, even if the QA system would have selected a correct answer without reformulations, a strong match with a reformulation as opposed to a weaker match with the original question can provide additional confidence in the correctness of the answer.

References

Agichtein, E., S. Lawrence and L. Gravano. Learning Search Engine Specific Query Transformations for Question Answering. WWW10, 2001.

Banko M., E. Brill, S. Dumais, J. Lin 2001. AskMSR: Question Answering Using the Worldwide Web In *Proceedings of the TREC-2001 Conference*, NIST. Gaithersburg, MD

Bikel, D., R. Schwartz, and R. Weischedel. 1999. An Algorithm that Learns What's in a Name.*Machine Learning-Special Issue on NL Learning* 34, 1-3.

[1] TextMap is the successor system to Webclopedia (Hovy 2001).

Fellbaum, C. Editor; G. Miller, preface 1998. MIT Press.
URL: http://www.cogsci.princeton.edu/~wn/

Hovy, E., L. Gerber, U. Hermjakob, C.-Y. Lin, D. Ravichandran 2001. Towards Semantics-Based Answer Pinpointing In *Proceedings of the HLT 2001 Conference*, San Diego

Lin, D. and P. Pantel 2001. Discovery of Inference Rules for Question Answering. In *Journal of Natural Language Engineering 7(4)* pp. 343-360.

Kwok, C., O. Etzioni and D. Weld. Scaling Question Answering to the Web. WWW10, 2001.

Radev, D., H. Qi, Z. Zheng, S. Blair-Goldensohn, Z. Zhang, W. Fan and J. Prager. Mining the Web for Answers to Natural Language Questions. ACM CIKM 2001.

Ravichandran, D. and E. Hovy 2002. Learning Surface Text Patterns for a Question Answering System In *Proceedings of the 40th meeting of the Association for Computational Linguistics*, Philadelphia, PA

Witten, I.H., A. Moffat, and T.C. Bell. 1994. Managing Gigabytes: Compressing and indexing documents and images. New York: Van Nostrand Reinhold.

TREC 2002 – WEB TRACK

"Automated Word Sense Disambiguation for Internet Information Retrieval"

Christopher M. Stokoe
University of Sunderland
Informatics Centre
St Peters Campus
+44 (0)191 515 3291
Christopher.Stokoe@sund.ac.uk

Prof. John Tait
University of Sunderland
Informatics Centre
St Peters Campus
+44 (0)191 515 2712
John.Tait@sund.ac.uk

Abstract

We describe an attempt to use automated word sense disambiguation to improve the performance of an internet information retrieval system. A performance comparison of term frequency verses word sense frequency was carried out, the results of which indicated no significant performance gains from using a sense based retrieval model instead of the traditional TF*IDF.

1. Introduction.

Several authors have observed that ambiguity in language [1,2] can have a negative effect on the performance of text retrieval systems. Over the past ten years a number of researchers [1, 3] have worked on trying to integrate word sense disambiguation (WSD) techniques into text based information retrieval (IR) systems in an attempt to eliminate ambiguity and increase effectiveness. These attempts have on the whole produced disappointing results with the exception of Shütze and Pederson[4] who achieved a relative 14% increase in precision using a combined sense and term based model over traditional term only strategies. However it is important to note that often work in this field has been difficult to assess due to a failure to effectively evaluate the accuracy of the disambiguation used. Additionally there is evidence to suggest that often these experiments were undertaken on small or unsuitable collections. To this end the authors identified the need to re-examine the possible effects of automated word sense disambiguation in text retrieval systems using more rigorous performance measures.

2. Hypothesis.

Given that studies [5] have identified short queries may benefit most from a disambiguated collection we set out to evaluate the performance of automated word sense disambiguation within a web search system by submitting a class C entry to the TREC 2002 Web Track. The aim of this experimental work was to assess the relative benefits of searching from a collection with reduced ambiguity in an attempt to identify whether the introduction of automated word sense disambiguation can produce more effective results. In order to achieve this, the authors attempted to run a base line experiment taking effective performance measurements for both the disambiguation and retrieval models to assure the validity of the work.

3. Experiment Methodology.

To gain an accurate assessment of the effects of incorporating word sense disambiguation we created two full text indexes of the .GOV corpus. Each document in the collection was parsed using www::parser perl library and the resulting output was catalogued as plain text in *index(a)* and then fed into the automated word sense disambiguation system and sense tagged (see section 4). The sense tagged version was then added to *index(b)*. The format of the indexes used was relatively crude due to time constraints and although this subsequently effected retrieval times this was considered irrelevant in the scope of our experimental goals. Total index time was 7Days and 4Hours for *index(a)* and 22days 3Hours for *index(b)*. The indexing was carried out on a 1GHZ Pentium 3 with 398Mb of memory running Linux.

Once the indexes had been completed two topic distillation runs were performed and submitted to NIST. These runs were designed specifically to contrast the performance of the word and sense index models.

4. Automated Disambiguation.

The disambiguation strategy used was a statistical system trained using the Brown1 part of Semcor1.6 which is distributed with WordNet. Semcor consist of a subset of the Brown corpus manually disambiguated against the sense definitions contained in WordNet. The main language features used by our disambiguation system were sense frequency from both Semcor and WordNet, idioms and co-occurrence statistics observed from Semcor. These techniques were combined based on their individual accuracy to provide a hierarchy by which to select the appropriate sense. Each was applied using a context window consisting of the sentence incorporating the target word to be disambiguated.

The algorithm used to perform disambiguation was relatively simple using a stepwise approach to move through the techniques until either finding an appropriate match or falling back to sense frequency. Techniques were applied in descending order of their individual performance as determined in previous experimentation.

The performance of the disambiguation engine was measured in Precision and Recall and evaluated using the Brown2 part of Semcor1.6. Brown2 consists of 86,412 sense tagged word instances representing a traditional all-words test collection. Results were encouraging with the system scoring 60.1% precision and 57.9% recall. Coverage of the test corpus in terms of words attempted was 96.32%. Overall the system performed above the current baseline for disambiguation systems established from the Senseval-2 literature [6].

5. Retrieval Technique.

The retrieval mechanism used in both runs was a Boolean "AND" search with the results being ranked based on Salton and McGill's TF*IDF [7] measure summed across all terms in a query. The queries were stop worded and a rudimentary form of stemming was incorporated. The first run (TDtfidf) was a base line run carried out using *index(a)* in order to asses the relative performance of our combined retrieval and topic distillation technique. Our second run (TDwsdtfidf) was carried out using *index(b)* with TF*IDF calculated using sense occurrence rather that term frequency.

Overall performance in terms of speed of query execution was poor; however this was to be expected given the simplistic nature of our index strategy. Average processing time per query was 37.3 minuets for the term frequency model lowering to 34.1 minuets for sense frequency.

6. Topic Distillation.

Our strategy for topic distillation was relatively simplistic but because of time constraints it was impossible to carry out more traditional approaches such as link analysis or frequency distribution. As such our technique involved a post processing task carried out over a query's results to identify multiple hits / instances of pages from the same site. Once multiple hits from a single site had been identified the system reduced their URL's to the lowest common point of agreement where there existed a page in the document collection. this we refer to as the topic root. This page was then returned with an aggregate of the combined weighting score of its constituent elements. The multiple occurrences were stripped from the results and replaced with the topic root page ranked appropriately. The rational behind this strategy was the hypothesis that the arrangement of a site across the directory structure of a web server could be used to effectively assess the best point at which to enter the site for a given query.

7. Results.

We submitted two Topic Distillation runs for evaluation the first TDtfidf used term frequency and the second TDwsdtfidf used sense frequency. Both runs were identical in terms of the number of documents / number of relevant documents retrieved. However an examination of the systems results indicates subtle differences in the rankings. Table 1 shows the average Precision (non-interpolated) and R-Precision figures for both runs across all 49 queries.

Table 1: Combined results for runs.

Run Tag	R-Precision	Average Precision (Non-Interpolated)
TDtfidf	0.0451	0.0211
TDwsdtfidf	0.0454	0.0211

R-Precision shows a small increase in the performance of the sense model (TDwsdtfidf) when compared to the term model (TDtfidf).

Table 2 shows a comparison of the Interpolated Recall Precision Averages. When examining the Interpolated Recall – Precision figures we see improved precision in the low recall range when using the sense frequency model.

Table 2: Interpolated Recall - Precision

Interpolated Recall	TDtfidf	TDwsdtfidf
At 0.0	0.2941	0.2952
At 0.1	0.0751	0.0760
At 0.2	0.0180	0.0181
At 0.3	0.0040	0.0038
At 0.4	0.0008	0.0008
At 0.5	0.0008	0.0008
At 0.6	0.0000	0.0000
At 0.7	0.0000	0.0000
At 0.8	0.0000	0.0000
At 0.9	0.0000	0.0000
At 1.0	0.0000	0.0000

From this we can see that using sense information helped to promote a small number of key resources.

8. Conclusion.

The main aim of this project was to assess whether automated word sense disambiguation could be used to improve retrieval effectiveness. Although the use of automated disambiguation did lead to a small (0.0003%) increase in R-Precision this is considered statistically insignificant and as such the overall results were disappointing. There are several possible explanations for this.

- Firstly, the Topic Distillation strategy used was weak and in many cases missed the optimum page to return. The technique tended to reduce a key resource to the highest possible entry point of a particular site.
- Secondly, although our WSD strategy tested strongly in terms of overall accuracy it relied heavily on WordNET'S frequency statistics for words that had not been encountered in training our system. This meant that if no training data was available for a word all instances would be assigned the same sense which effectively failed to reduce any ambiguity from the corpus. Therefore increased training data could potentially lead to performance benefits.

Despite these problems it is important to note that many previous attempts to use automated disambiguation in IR have significantly reduced the performance of IR models such as TF*IDF. Although the performance gains we achieved were minimal the 39.9% error rate of our disambiguation methodology did not have a negative impact on retrieval performance. This runs contrary to the findings of Sanderson, 2000 [1] however further investigation is needed to assess exactly how much ambiguity was removed from the .GOV corpus.

9. References.

[1] Sanderson, M. 2000. "Retrieving with good sense" in Information Retrieval Vol 2, No 1 Pp 49 – 69.

[2] Kowalski, Gerald; Maybury, Mark, 2000. "Information Storage and Retrieval Systems Theory and Implementation." Kluwer, Pp 97.

[3] Vooehees, E. M. (1993). "Using WordNet to disambiguate word sense for text retrieval" Appeared in proceedings of ACM SIGIR Conference (16): Pp 171-180.

[4] Shütze, H; Pederson, J. O. 1995 "Information Retrieval Based on Word Senses" in Proceedings of the Symposium on Document Analysis and Information Retrieval 4 Pp 161 -175.

[5] Krovetz, R; Croft, W. B. 1992. "Lexical Ambiguity and Information Retrieval" in ACM Transactions on Information Retrieval Vol 10 Issue 1.

[6] Edmounds, P; Cotton, S. 2002 "SENSEVAL-2: Overview" in Proceedings of the Second International workshop on Evaluating Word Sense Disambiguation Systems.

[7] Salton G; McGill, M.J. 1983 "Introduction to Modern Information Retrieval" New York: McGraw & Hill.

Augmenting and Limiting Search Queries

Elaine G. Toms, Luanne Freund, and Cara Li
Faculty of Information Studies
University of Toronto
Toronto, Ontario M4W 3V8 Canada
{toms, freund}@fis.utoronto.ca; cara.li@utoronto..ca

Introduction

Web queries tend to be significantly shorter and less complex than queries used in earlier types of information systems (Jansen & Pooch, 2000; Lawrence & Giles, 1999; Spink et al, 2001). Yet, there is general belief that enriched queries and query reformulation will lead to improved results (Belkin et al, 2001). In our research we are examining the sorts of tools that could assist with the creation of enriched queries and in turn improve the search process and the user's search experience.

In the work reported here we assessed the use of two types of tools: one to assist the user in targeting and, thus, restricting the query, and a second one to assist in augmenting the query. We speculated that certain types of tools are more useful for certain types of information tasks. In particular we targeted the standard informational request in which a suitable response could be culled from many different Web pages, and secondly, the 'know-item' task, in which a specific Website exists to solve the problem. We anticipated that the tool to enable query augmentation would be more useful for informational tasks, as it would encourage amplification of the query to contain many more words and phrases that represent the information task. On the other hand, 'know-item' searches suffer when the exact title or some exact content is not known in advance and the limiter would enable more focussed searches.

Our hypotheses were:
1) Restricting a search to selected Internet domains would improve results for "know-item" information problems
2) Amplifying the query were additional keywords would improve results for information oriented information problems.
We tested these hypotheses in a within-subjects experiment using a novel information retrieval experimentation system prototype, called WiIRE which was run on the Web and required no researcher-participant communication.

Methods

Participants

Twenty-four participants (15 females and 9 males) took part in the study. About half were under 28, the majority of which were young – between 18 and 22; all but one was under 43. Most had either an undergraduate degree or were currently enrolled in one. All have used the web for two to four years. Forty-six per cent use a computer for 11 or more hours a week and use the web for an almost equal additional amount of time. About half use a computer daily for home, work or school purposes.

About 75% use search engines almost daily and, in general, claim to find what they are looking for. Daily, about 90% use search engines for work/ school related project and about 67% for entertainment. The search for information in other areas was very different: very few search for shopping (13%), travel (8%), health (21%), or government information (17%) on a daily basis, and about 25% never search in for those types of information. Because the data source used for this study was the United States government websites in the *.gov* domain, and the participants were Canadian, we also inquired about their previous search experience in this domain and their familiarity with United States government structure and agencies. About 25% searched monthly in the *.gov* domain, but no one searched more frequently than that. Approximately 70% have never (knowingly) searched in the *.gov* domain. Nearly two-thirds confessed to being unfamiliar with the US government structure and agencies. Thus our participant group was unfamiliar with the US government and infrequent searchers of its web pages, but they were well acquainted with searching the Web in general.

Search Engine

The Panoptic search engine(see http://www.panoptic.com) developed by Australia's National Research Agency (CSIRO) and the Australian National University was used in the study. Panoptic is a probabilistic search engine which accepts natural language query input as well as advanced query operators. The search engine was operating on a static collection of United States government web pages crawled in January 2002. The document collection contains 18 gigabytes of data from 1 million web pages. This particular set-up was created in an attempt to do a controlled experiment in a Web-like laboratory.

Tools

To test our concept of enhancing the query creation process, we developed two types of interface tools tailored to the data source used in this study.

a) *Limiters: Agency Locator and Acronym Identifier*

Both serve to limit the search. We know from experience with government information that finding the right department and sub-department is often key to the right information. The intent of these tools was to enable the searcher to restrict the search to a particular agency.

! *Agency Locator* contains a select group of government agencies, departments and offices, selected from the Louisiana State University library's hierarchical directory of Federal government agencies (www.lib.lsu.edu/gov/tree). This list was presented in a collapsible tree format. Selecting one or more branches on the tree restricted the search to the web domain of that particular agency or group of agencies. The list was arranged by broad subject areas.

! *Acronym Identifier* contains a select list of common acronyms used in/by government agencies. Because the URLs for *.gov* sites contain agency, program, office, etc. acronyms, the contents of the URL provide valuable clues about an item on the results list. The acronym list could be used both to restrict the search to that particular group(s) and also to help interpret the URL, providing value-added information to the user. In in-house tests on the web in general, we have found that users interpret the content of URLs and use that information in making choices from a results list. In this case, we wanted to simplify that process. The tool worked as an alphabetical look-up list.

In each case, we used Panoptic's domain specific search feature to limit by specified domain(s).

b) Augmenter: Keyword Finder

The *Keyword Finder* was devised to suggest keywords, as well as give the user some notion of the kind of words that exist in the index and how they are represented in the database. We created this tool from Panoptic's 10-million word keyword index. Part of our problem was reducing Panoptic's index to something that was humanly digestible. Using a step by step process that used a combination of simple heuristics and WordNet, we reduced the list to around 20,000 words which were presented in an alphabetic list that also contained the keyword frequency information.

Interface

The interface to the Panoptic system was embedded in a testing environment (see last section in Methods). The interface was built using a three-frame design. The top frame was used only for experimental purposes. The bottom two frames contained and controlled access to the Panoptic Search Engine. The query input box was positioned on the left side while the query limit/augment tools were contained in the right frame. The tool in place depended on the experimental condition. When search results were returned, the search page on the left was refreshed to now contain both the query input box as well as results. Twenty hits were displayed per page. When a URL was selected from the results, a new 'website' window open to the right containing the website. Thus the website could be viewed in context with the results page, eliminating the need for constant backward movements. Each successively selected URL was presented in the website window.

Queries were processed with stop words filtered. Conventionally Panoptic presents results in tiers: the first tier contains all of the query words while documents which partially match are presented in subsequent tiers. If queries are very long, the keywords are processed in order of decreasing rarity and the most common words are ignored.

One of the following two options was present at the interface:
a) In the Limit condition, all selections from the Acronym Identifier and Agency Locator inserted the domain name for a domain restricted search to the Limit box in the Query frame. Subsequent additions were aggregated in this box. When the search was executed, the Limit box contents were appended to the query string and served to restrict the search to URLs containing the domain string.
b) In the Augment condition, all selections from the Keyword Locator were added to the query box. The resulting query string contained both manually entered terms and terms added through use of the Keyword tool. In this condition there was no Limit capability.

Under both conditions, we used a "bookbag" metaphor as a means for participants to indicate which pages from the results list were useful for the given task. Participants were asked to mark the checkbox beside relevant results and click on the "Add to bookbag" button. The URLs, the title of the webpage, and rank on the hitlist of each selection was stored in a database together with details of the participant and task.

Variables

The independent variables which were assessed using a series of metrics were:
1. Interface:
   a). Limit: the Acronymn Identifier and Agency Locator tools
   b). Augment: the Keyword Finder tool
2. Task Objective:
   a). Document: objective of the search is a 'known-item' – a Web page
   b). Information: objective of the search is a set of pages on which the solution will be found.

Information Problems

The eight information problems used in the study were devised by the Interactive Track, 11th Text Retrieval Conference specifically for use with the *.gov* data source (see Table 1). Four information problems were randomly assigned to each of the 24 participants so that, overall, twelve people addressed each problem. Problems were characterized according to type: a) whether they required Information or Documents to respond to the problem as defined above. Those of type Information needed certain facts or details, while those defined as type Document required a specific web page or pages to respond to the information problem.

#	Information Problems	Type
1	You are travelling from the Netherlands, and want to bring some typical food products as gifts for your friends in the United States. What are three kinds of food products from the Netherlands that you are not allowed to bring into the US?	*Information*
2	You are concerned with privacy issues related to electronic information and would like to know what laws have been passed by the US Congress regarding these issues. Identify three such laws.	*Document*
3	A friend has a private well which is the family's only source of drinking water. Locate a US publication, which contains guidelines for the maintenance of safe water standards for private well use.	*Document*
4	You are not sure about the safety of genetically engineered foods, and would like to find more information and research on this topic. Name four potential types of safety problems that have been raised.	*Information*
5	You are interested in learning more about what measures the US government has taken since 2001 to prevent Mad-Cow Disease. Identify three such measures.	*Information*
6	Name/find three research programs/projects that investigate the treatment/causes of	*Information*

#	Information Problems	Type
	dwarfism.	
7	You are planning a cycling expedition along the Silk Road in Central Asia. Find a website that is a good source information about health precautions should you take.	*Document*
8	You are planning to travel to the northeast territories of India and wonder if there are any problems/restrictions for tourists. Find a website that is a good source of information about such problems/restrictions.	*Document*

Table 1. Information Problems Used in the Study

Experimentation Environment – *WiIRE*

To manage this experiment we created a novel testing environment called WiIRE, the Web Interactive Information Retrieval Experimentation System. All experimental tasks and participant communications were contained within a web-based system. All data, from agreement-to-participate (the traditional consent form) to questionnaire, were captured into a database, or in server logs. For more details about this system, see Toms, Freund and Li (2002).

Procedure

Initially, each participant was given a prelude to the study followed by an introduction to the experimentation system, which started with the consent form. On agreement, this led to a demographics and search experience questionnaire and an overview of the study. The pattern beyond that point included an introduction to the new tool and practice time to use it. When participants indicated readiness to begin a task, they were assigned by the system a search topic. For each topic, participants completed a pre-task questionnaire, searched for information on that topic, and then completed set of post-task questions that contained both closed and open-ended questions to capture some of the data that would normally have been handled via interview. This process was repeated with the second interface and the last two search topics. At the end, participants were assigned a final summary questionnaire.

Data Analysis

Most of the data was captured to an MS-Access database or in server logs. Because participants came to our lab, we collected data using *WinWhatWhere*, a client-side transaction log file. The MS-Access database was converted to an SPSS data format. The URLs of all sites viewed was retrieved from the client-side transaction log files. The queries submitted by participants were retrieved from the server logs. In addition, the server logs and the client-side transaction logs were blended to retrieve time data and selected process data.

Results

The purpose of this study was to assess the effectiveness of two tools to aid query creation. To do so we examined several factors that affect the search process: 1) what the participants brought to the process – their knowledge and experiences that may affect outcomes; 2) effectiveness of the interface to supported the user in the query creation/reformulation processes and in the interpretation of results; 3) the performance of the system: how well it interpreted the query and provided a suitable set of results; 4) outcomes: how well the user and the system were able to accomplish the task, and how well the user perceived the process. The metrics were based on participant assessment, and on objective observation collected via logs, the outcome, and on independent assessment.

Pre-Task: Prior Knowledge and Experience of Participants

No participant had searched any of the information problems used in this study prior to this study. Participants indicated a lack of expertise in the topic area: 70% were unfamiliar with the topic and a further 21% were somewhat familiar, while 84% indicated having no or little expertise in the topic area at all. About 50% believed that there was information available for each information problem while 25% believe that was little information on the topic, and a similar proportion (50%) of participants believed that the information problems would be easy to solve. In summary, while participants had a lot of search experience, but no formal training (as described earlier under participants), and had no knowledge or expertise with the topics, they believed that there was information out there, but it might not be easy to find. Furthermore, there were no significant difference in perceptions according to task objective (F(1.4)=.934, p=.448) or to interface (F(1,4)=1.468, p=.219).

Analysis of the Queries

Participants created 395 queries, an average of 4.1 queries per task. The queries were evenly divided between the Limit and Augment conditions. But in the task objective condition, 54% were created for Information problems versus 46% for Documents. There was an interaction of task objective and interface. Fewer queries were created for Documents than for Information in the Keyword condition.

Each query was composed of, on average, 4.3 keywords, ranging from one keyword to fourteen. Of these Panoptic used 3.7 in the retrieval process. There was no interaction of task and interface (F(1,391)=.178, p=.673). There was also no main effect of task (F(1,391)=.95, p=.482), but there was one of interface (F(1.391)=6.935, p=.009). Participants in the Augment condition created significantly larger queries (4.6) than those in the Limit condition (4.0).

As previously described, each information problem was represented as a sentence or statement to participants. Typically one keyword per query did not come directly from the assigned information problem, but was devised by the participant. There was no interaction of task objective and interface concerning the number of words in those queries that did not also appear in the original information problem statement (F(1,391)=.144, p=.704), but there were main effects of both task and interface. Those in the Limit condition added significantly fewer words to queries than those in the Augment condition (F(1,391)=27.370, p<.0001). Similarly, those searching for Information added significantly more keywords than those searching for Documents (F(1,391)=15.260, p<.0001).

Twenty per cent were simple keyword queries, while 52% contained a phrase. A further 28% contained a statement or question. About 34% of all queries created for Information problems were simple keyword queries compared to 4% for Document tasks. Seventy-seven per cent of all Document queries contained a phrase whereas only 33% of the Information queries had phrases. In the interface condition, between 50% to 60% were phrase queries, with the remainder about evenly split between keyword and sentence formatted queries. This pattern was retained regardless of interface used.

The content of the queries was further analyzed both from a linguistic perspective and for the use of syntax. Eighty per cent contained no advanced syntax, e.g., use of Boolean or "+" and "-." The proportion was the same in both task and interface conditions.

Over the course of a search, 75% of the second and subsequent queries for the same task were modified in some way: 18% were expanded in scope, while 9% were reduced in scope and 30% were completely revised. The remainder were re-entered. Twenty-two per cent of those in the Augment condition were expanded compared to 13% in the Limit condition. In the task objective condition, 35% of those seeking Information were totally revised compared with 25% in the Document conditions.

In addition, we examined how the tools were used within the task type. There were no significant differences by either the number of times the Limit tool was used or by the number of agencies appended to the query by type of task objective. On average the Limit tool was used about twice per information problem for a total of two agency URLs appended per task. The Augment tool was used on average once per query; a new keyword was added to a query about once for every two information problems. Of these keywords about the same number were new to the information problem. That is, the words did not already appear in the information problem statement.

Search Engine

Aboutness is a measure of how well the page fits the task. This is not a user-centred relevance judgement, but an objective assessment of the search engine's performance. Each web page examined by the participants was assessed by an independent expert for each information problem. Aboutness is partially a user selection, but more so, evidence of the a search engine's ability to deliver what the user has requested. Users can only select from what the search engine delivers, but then must make judicious choices from those items. The rules are listed below:

Rating	Meaning	% pages
5	pages directly related to the topic and containing clear info on the topic	14%
4	pages that provide some information that is related, or leads directly to the answer	17%
3	pages that about the topic but may be broader or narrower that the topic	26%
2	tangentially related but not really in the topic area	14%
1	pages that are clearly not about the topic at all	29%

The percentage in the third column contains the proportion of URLs examined that were assessed to be at that rating level. The average aboutness for all pages examined by participants was 2.7; 57% of the web pages examined were related or somewhat related to the topic being searched. The differences within each condition was insignificant at each rating level. In the interface condition, the percentage of pages at each level was about evenly split. This was not the case in the task objective condition when the trend was to a larger proportion of the pages rated a "4" or "5" in the Information condition. Thus pages retrieved in the Information condition tended to have a larger number of pages rated as on or relevant to the topic than those in the Document condition.

About 30% of the first URLs examined had an aboutness rating of "1." All second and subsequent URLs selected had a positive change (or no change) in the aboutness rating. In general, 27% of the URLs represented a higher aboutness rating than the one examined prior to it. There appears to be no difference by task objective or interface.

Of the 837 items examined, 400 were added to the bookbag. Presence in the bookbag signifies that participants considered the item to be useful in responding to the information problem. On average, participants inserted, per task, about 4.5 items into the bookbag while examining about 6.2 URLs. Forty-seven per cent of those added to the bookbag were rated a 4 or 5 on the scale presented above; 10% were rated a "1", or clearly not about the topic at all. Thus about half of the items in the bookbag were related to the information task.

The items contained in the bookbag also captured the rank of the item on the results list. The range of items viewed varied from a rank of 2 to 100 with the average being 61 (S.D. = 27.5). Twenty items were presented on each results page which means that participants examined up to the first five pages of results. The proportion that appeared on each page are listed below:

Page 1 (rank 1-20):	7.5%
Page 2 (rank 21-40):	18%
Page 3 (rank 41-60):	26.3%
Page 4 (rank 61-80):	18.2%
Page 5 (rank 81-100):	29%

There was no interaction of task objective and interface (F(1,80)=.650, p=.423) and no main effect of interface (F(1,80)=2.052, p=.156), but there was a main effect of task type (F(1,80)= 4.078, p=.047). Those in the Information condition had an average lower rank than those in the Document condition. Thus the search engine appears to be outputting better pages for for those classified as informational than for .know-item searches.

Outcome

Completeness of the Task

Completeness assessed the set of URLs examined by a participant for a task to answer the question: how completely could the information problem be resolved with the web pages collected in the bookbag? The following rules were used:

Rating	Rule
1	0% of the problem was answered/responded to
2	about ¼ of the problem was answered
3	about ½ of the problem was answered
4	about ¾ of the problem was answered
5	100% of the problem has been answered

Thirty-three per cent of the tasks were consider "complete" as verified by an external expert, while 42% were considered incomplete, that is the former were rated 5 on the scale while the latter were rated 1, and the other ratings were fairly evenly distributed between 7 and 9 per cents. Overall, about half a problem was complete. While there appears to be no differences by interface, more tasks in the Information condition (40%) were complete compared with those in the Document condition (25%) (Chi-Square= .044, df=4).

User perception: process

In general, about 52% of participants found the topics very easy to search, a proportion that is increased to 70% if the median "Somewhat" is also included. Approximately 42% claimed that they had more than enough time to do the search and a further 29% said that it was "Just right." About 76% indicated that their previous knowledge did not help with the search (which was not surprising considering the initial indication of their knowledge and expertise in the topic areas). About 47% indicated that their search experience was satisfying, while a further 21% were "neutral." For each of these measures there was no main effects or interaction of the factors.

User perception: outcomes

Participants were tri-modal about what was learned in course of doing the searches. About 1/3 learned very little and 1/3 learned a lot about the topic. In this case there was a borderline interaction of task objective and information (F(1,88)=3.723, p=.057). More participants in the Limit condition felt that they learned a lot while doing Document searches. The exact reverse was true for those in the Augment condition. About 51% were satisfied with their search results which is increased to 67% if the median "Somewhat" is included. Fifty-seven per cent indicated they would likely recommend their search results to a friend who was seeking the same information and a further 17% indicated they "maybe" would do so. About 61% were certain that they found answers to the search topics. For each of these measures there were no main effects of task objective and interface and no interaction.

Analysis

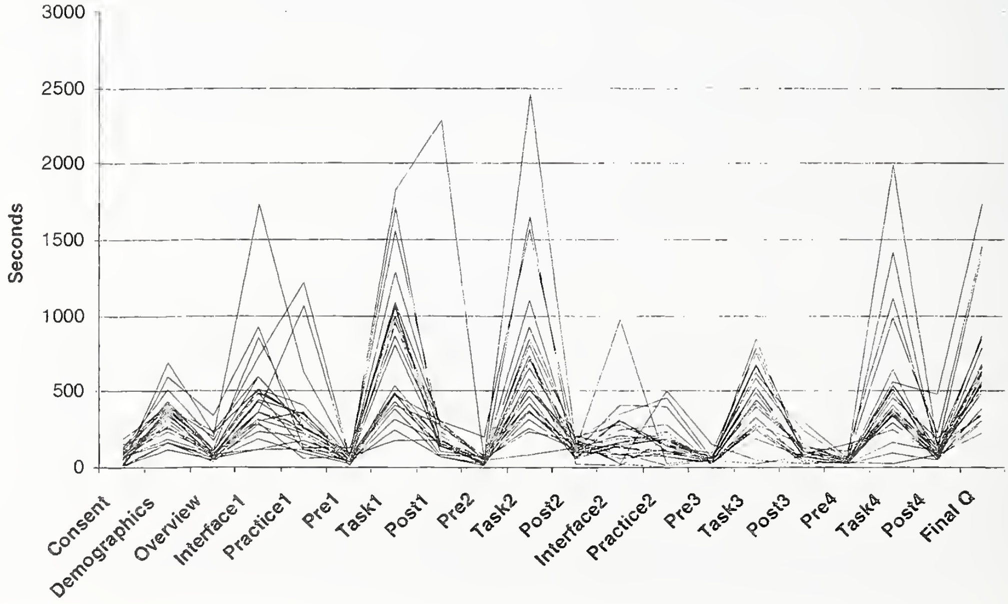
This research assessed the effectiveness of two types of tools, one to restrict the search to Internet domain and one that enabled amplification. We anticipated that the Limit tool would be more useful for Document tasks, and the Augment tools for the Information task. Results are not clear cut and we cannot conclude that our hypotheses were supported. For many metrics there were no anticipated interaction effects from which we can conclude that one tool is more suited to a certain type of task. Surprising in the results was the high rank of relevant hits when compared with the typical Web search result in which users rarely go beyond the first couple of pages of hits. When combined with the aboutness and completeness rating, we are suspicious about both the ability of the search engine to find good pages. In addition, participants were unfamiliar with the topics to be searched and with US government information and their ratings and personal perception at the end illustrates those issues.

Participants in the Augment condition created significantly longer queries than those in the Limit condition.

suggesting that the Keyword tool, either directly or indirectly affected the query. In data assessed to date, the use of the tool does not suggest that it affected ability to complete. The Agency finder was used by the most participants (21) and is the only tool that received an above average rating on a scale of one to seven. The keyword tool was the least transparent to users, and most of the comments indicated that participants either did not understand how to use it: "I don't understand the use of the keyword finder when you can just type the word by typing it in the box. (P21)", or found it too unwieldy to use: "It was very time consuming to scroll through the list... (P11)". From the comments, it seems likely that most participants did not understand how to apply this data to query formulation. Participants found the Agency tool "a great help in narrowing down the search area (P10)". Participants noted that it suited the government domain, but that it might not be helpful "in all subjects of interest (P14)". The low level of use of the Acronym tool seems to be related to its more specific application. The general thread of the comments was that it looked potentially useful, but the need for it did not arise: "I just didn't need to use it in this instance, however, I think I'd use it right away if I didn't know the agency (P3) ". Thus, despite their overall performance participants understood the intent of the tools, except perhaps the Keyword Finder. So why did those tools not help? Prior to the start of the study, we were able to identify by limiting by agency more useful items in selected tasks, yet the participants were seemingly unable to do the same. Likely the problem is in its implementation.

WiIRE – the Web Information Retrieval Experimentation System prototype

WiIRE enabled the processing of participants in three groups, rather than the usual one-on-one, two-hour sessions; this resulted in considerable efficiencies in data collection. Form-based data was collected in a database while process-based data such as time and queries were acquired from web server logs. The other data critical to this study that could not be collected in the server logs was the URLs visited. For this we relied on a client-side transaction logs, a tool that would not be available to us with remote participants. However, we did incorporate a 'bookbag' feature at the hitlist level, which stored items and their rank in a 'bookbag.' This proved to be too cumbersome, as participants had to go back to the hitlist to add items to the bookbag, a task we plan to add at the page level in the next version.



The figure above illustrates the process taken by a participant in WiIRE. Noteworthy about this figure is the amount of time spent doing each activity in the experiment from reading the consent form to doing the final questionnaire. The bolded heavy black line is the average time for each activity. The most interesting factoid is the amount of time taken to do each task which decreased significantly over the course of the study. (In the figure, tasks are in order of completion and not by topic number.) This raises significant questions about the length of time that should be allotted for human experiments and the number of tasks that should be assigned before the process becomes onerous.

Conclusions

In this study, a modified search interface containing two types of tools designed to help with query creation were tested using the Panoptic search system. Overall results were inconclusive, as few participants could respond completely to a search topic and the pages examined were mostly not about the topic assigned, at least as determined by an external assessor. Part of the problem is likely due to the specialized nature of the database – the US government web pages, and to the lack of knowledge and experience of the participants with this type of data. Although the participants rated the experience slightly above average in their personal assessments, one has to question the effectiveness of the tools as well as the search engine. In addition a new web-based experimentation system was tested. In the lab it enabled bulk processing of participants without degradation in data quality. Noteworthy about its use was the significant drop in time taken to do a search task as the experiment progressed.

References

Belkin, N. et al. (2001) Rutgers' TREC 2001 Interactive TREC experience In: Text Retrieval Conference 2001, NIST, DARPA, ARDA, 452-461.

Jansen, B. J. & Pooch, U. (2000) A review of Web searching studies and a framework for future research. Journal of the American Society for Information Science and Technology 52 (3), 235-246.

Lawrence, S. & Giles, C.L. Searching the web: general and scientific information access. IEEE Communications Magazine (January1999), 116-122.

Spink, A. Wolfram, D., Jansen, B.J., & Saracevic, T. Searching the web: the public and their queries. Journal of the American Society for Information Science, 52, 3(2001), 226-234.

Toms, E.G., Freund, L., & Li, C. (2003). WiIRE: a Web Interactive Information Retrieval Experimentation system prototype. (Submitted)

Statistical Selection of Exact Answers (MultiText Experiments for TREC 2002)

C. L. A. Clarke G. V. Cormack
G. Kemkes M. Laszlo T. R. Lynam E. L. Terra P. L. Tilker

School of Computer Science, University of Waterloo, Canada
mt@plg.uwaterloo.ca

1 Introduction

For TREC 2002, the MultiText Group concentrated on the QA track. We also submitted runs for the Web track.

Building on the work of previous years, our TREC 2002 QA system takes a statistical approach to answer selection, supported by a lightweight parser that performs question categorization and query generation. Answer candidates are extracted from passages retrieved by an algorithm that identifies short text fragments containing weighted combinations of query terms. If the parser is able to assign one of a predetermined set of question categories to a question, the system employs a finite-state pattern recognizer to extract answer candidates. Otherwise, one- to five-word n-grams from the passages are used. Our system assumes that an answer to every question appears in the TREC corpus, and it produces a NIL result only in a few rare circumstances. Despite the simplicity of the approach, our best QA run returned correct answers to 37% of the questions.

Our basic question answering strategy is an extension of the technique we used for both TREC 2000 and 2001 [5]. In past years, our system ranked individual terms appearing in retrieved passages and selected 50-byte responses from the passages that included one or more of the highest ranking terms. Since exact answers are required for TREC 2002, much of our effort this year was focused on the extension of this technique to multi-term exact-answer candidates.

Last year, a novel feature of our QA system was the use of commercial Web search services to reinforce answer candidates. This year we reduced our dependence on these commercial services by generating our own collections of structured and unstructured data for use in question answering. The structured data collection consists largely of tables containing answers to questions of frequently occurring types, such as the names of capital cities, the names of world leaders, and the names of baby animals. The unstructured data collection consists of a terabyte of Web data gathered from the general Web in mid-2001. For comparison, half of our submitted runs use a commercial Web search service (Altavista) in addition to our own collection.

As a supplement to our basic question answering strategy, we developed an "early answering" strategy using our structured collection. Under this strategy, if a question can be answered directly from the structured data, the problem is reduced to one of answer justification, in which our system attempts to locate a document in the TREC corpus where question and answer keywords appear in close proximity. If no acceptable justification can be found, or if the question cannot be answered with the structured collection, our basic question answering strategy is invoked. This combination of two strategies — a strategy that searches a federated collection of structured data with a statistical strategy that searches a large collection of unstructured text – is essentially the approach to question answering advocated by Lin [9]. Our experimental runs examine the impact of our early answering strategy. Half our submitted runs use the strategy and half do not.

In the next section we provide an overview of each of the main components of our QA system, with a particular focus on the components which are new to our system for TREC 2002. Section 3 gives the results of our QA track experiments. For the Web track we submitted runs that took advantage of anchor text and link information in addition to document content. In section 4 we describe the approach used for our Web track experiments and discuss the results.

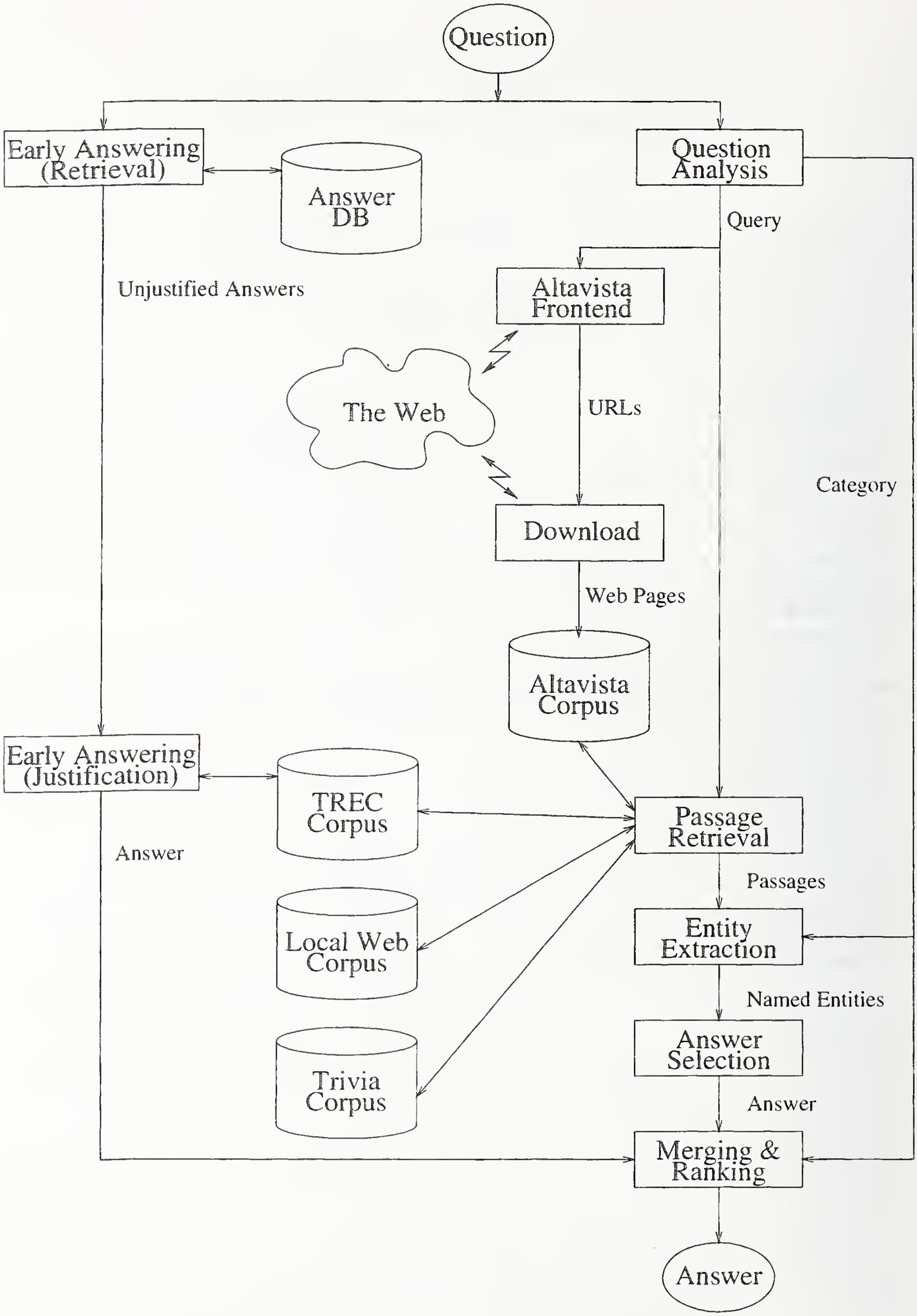


Figure 1: QA System overview.

2 The MultiText QA System

Figure 1 provides an overview of our QA system, showing the processing steps from question to answer. Not all the paths through this diagram are used in all runs. Half of our runs omit the Altavista path through the center of the diagram, and half of our runs omit the early answering path down the left.

2.1 Question Analysis

The question-analysis component takes a natural language question as input and yields a set of terms amenable for input to the passage retrieval system. Question analysis also yields answer categories used by the entity extraction component.

To achieve its objectives, the question analyzer looks for textual elements characteristic of some question form. To this end, we use a context-free grammar generating the forms of interest, and find the most likely derivation of a question using a probabilistic version of Earley's algorithm. The context-free grammar has been augmented to form an attribute grammar and the attributes are evaluated based on the most probable derivation.

Query generation has changed little from TREC 2001 [3] and is based primarily on part-of-speech and quoted-string attributes. Queries are term vectors. Our passage retrieval system provides a rich language for term expressions that includes exact matching of words and phrases, matching under stemming, matching of term disjunctions, and other less-standard operators. For query generation, part of speech is used to determine the role of each word as a query term. Nouns, adjectives, and adverbs are used directly as query terms. Verbs are stemmed if they are regular, and expanded if they are irregular. Articles and prepositions are discarded unless otherwise specified in the grammar rules. Quoted strings are used directly as query terms, as are the individual words contained in quoted strings.

Along with a query, the attribute evaluator outputs a set of global attributes and binary relationships among words in the question. This information was processed further using Prolog. As an example, the parser output for TREC 2001 question 1383 is shown in figure 2.

Global attributes are expressed as the unary relations qno for question number, inst for instance-of and so on. The binary relationships are expressed as 4-tuples:

e(*qno*, *source*, *sink*, *rel*),

where *qno* is the question number, *source* and *sink* are words in the question, and *rel* is the relationship between *source* and *sink*. For example,

```
e(1382,"shepard","make","subj")
```

indicates that "shepard" is the subject of "make", and

```
e(1383,"flight","make","obj")
```

indicates that "flight" is the object of "make". More importantly, we see that "spacecraft" is the object of an adverbial phrase modifying "make", and that "what" (the question word) modifies "spacecraft". Our Prolog analyzer is able to deduce from these relations that the answer being sought is the spacecraft involved in Shepard's flight.

We use WordNet to determine if a word named in the instance-of attribute is a person, place, thing, etc. and use this information in assigning the question a category. We use word relationships as further evidence in determining the category (e.g. authors write books, inventors invent inventions, and so on). Our original intent was to use the analysis to generate better queries and to aid in recognizing answer contexts. Time permitted us to use the analysis only for categorization.

2.2 Passage Retrieval

We have developed a passage-retrieval algorithm for question answering that can identify small text fragments that cover as many question terms as possible. This retrieval algorithm has been applied in all our TREC question-answering experiments to date. A detailed description of the algorithm may be found elsewhere [3, 5].

Unlike most other passage-retrieval algorithms, ours does not retrieve predefined document elements, but can retrieve any document substring within a corpus. Usually, these fragments are considerably smaller than the documents that contain them. The score of a fragment depends on its length, the number of question terms it contains and the relative weight assigned to each of these terms.

Fragments are often only a few words in length, and may cover only the question terms. To provide the context necessary for entity extraction and answer selection, each fragment is expanded by n words on each side, and this larger passage is retrieved from the corpus. The location of the original fragment within each larger passage is marked as a "hotspot". The answer-selection algorithm takes into account the location of answer candidates relative to this hotspot.

```
qno(1383).
inp(1383,"In in what spacecraft did U u S s astronaut Alan alan Shepard
        shepard make his historic 1961 flight").
cat(1383,"what").
cat1(1383,"what").
e(1383,"what","spacecraft","adj").
e(1383,"in","spacecraft","prep").
e(1383,"alan","shepard","adj").
e(1383,"astronaut","shepard","adj").
e(1383,"us","shepard","adj").
e(1383,"1961","flight","adj").
e(1383,"historic","flight","adj").
e(1383,"his","flight","adj").
e(1383,"flight","make","obj").
e(1383,"spacecraft","make","adv").
e(1383,"shepard","make","subj").
```

Figure 2: Parser output for TREC 2001 question 1383.

Passages are retrieved from at least three — and in some cases four — corpora. The first of these is the TREC corpus itself. The second corpus is our own local terabyte Web corpus. The third is a small 27MB corpus containing 330,000 trivia questions and answers, with each question/answer pair indexed as a separate document and treated as unstructured text by the passage-retrieval component.

Half of our runs use a fourth corpus generated by querying the Altavista search engine. In these runs, we download the top 200 documents returned by Altavista to form a small question-specific corpus. Since its contents are biased by the query, term statistics from the TREC corpus are used during passage retrieval from this small corpus.

The top 20 passages are retrieved from the TREC corpus, the top 40 from the local Web corpus and up to 10 from the trivia corpus. In runs using an Altavista corpus, the top 40 passages are retrieved. The system merges the retrieved passages and passes them to the entity-extraction component.

2.3 Entity Extraction

An entity-extraction component, with responsibility for identifying answer candidates, is a major addition to our QA system for TREC 2002. Our overall approach to entity extraction and answer selection is similar to the "n-gram mining" technique described by Dumais et al. [8]. An answer candidate is always a word n-gram appearing in a retrieved passage. If the question-analysis component assigns a category to a question, simple pattern matching is used to extract n-grams corresponding to the category. But when a category cannot be assigned to a question, n-grams of one to five words within a fixed window of the hotspot become the answer candidates. The entity extractor records the passage and document in which each candidate appears, and its location relative to the hotspot. All this information is passed to the answer selection component.

The primary purpose of the entity extractor is to eliminate unacceptable or unlikely answer candidates from the set of all n-grams. Thus, we prefer to retain questionable candidates and rely on the answer-selection component to give a low score to spurious candidates. As an example, for the PERSON category the entity extractor will accept any capitalized word surrounded by uncapitalized words, provided that it is not a stopword or question term and provided it appears capitalized in the corpus more than 50% of the time.

Pattern matching is achieved with a tool that allows the results of finite-state matching to be merged, filtered and cascaded, similar to the tool described by Abney [1]. The tool incorporates an algebra for structured text search [4], which, along with other capabilities, allows the context of a match to be considered. Finite-state automata may be specified by regular expressions or by lists of terms, such as the names of countries or states. In addition to finite-state automata, hand-written matching code may be called from the matching tool as needed.

A total of 48 categories are matched; a full list is given in figure 3. Most of the categories are self-explanatory, and many are standard in

AGE	AIRPORT	ANNIVERSARY	AREA
BIRTHSTONE	CODE	COLOUR	CONSTELLATION
CONTINENT	CONVERSION=unit0,unit1	COUNTRY	CURRENCY
DATE	ELEMENT	LAKE	LARGE
LENGTH	LONG	MASS	MEASURE
MONEY	MONTH	MOON	MOUNTAIN
NATURAL	NUMBER	OCEAN	PERSON
PHONE	PLACE	PLANET	PROPER
PROVINCE	QUANTITY=unit	RATE	RIVER
SEASON	SPEED	STATE	TEMPERATURE
THING	TIME	URL	VOLUME
WEEKDAY	YEAR	ZIPCODE	ZODIAC

Figure 3: Question categories.

question answering systems (e.g. DATE, CITY, PERSON, TEMPERATURE). A few (AIRPORT, BIRTHSTONE, SEASON) are inspired by previous TREC evaluations. Two categories, CONVERSION and QUANTITY are parameterized by units. After matching, we normalize candidates corresponding to time, measurement, and numeric categories to standard formats to assist the answer selection process.

Matching of the generic proper name categories (PROPER, PERSON, PLACE, THING) is a two-stage process that depends on corpus statistics. In the first stage, we identify lexically acceptable candidates using a longest-match approach. For example, the first stage would identify only the string "U.S. President Bill Clinton Thursday" as a candidate for the PERSON category in the passage:

```
...U.S. President Bill Clinton
Thursday proposed a five-year,
over 6 billion U.S. dollar package
to raise the ante on his nearly
fulfilled pledge to put 100,000 new
officers on the beat nationwide...
```

The second phase uses corpus statistics to propose substrings of the first-stage candidates as additional candidates. To generate these corpus statistics, we applied the longest-match patterns for the generic proper name categories to a concatenation of the TREC 2001 and 2002 QA corpora and recorded a count of each matching string. Any substring of a first-stage candidate that appears more frequently than the candidate itself is proposed as an additional candidate. However, single terms are not proposed if they are capitalized less than 50% of the time in the combined corpus. In the example above, the strings "U.S.", "Bill Clinton", "U.S. President Bill Clinton" and "Thursday" would be proposed as additional candidates. but not "S. President" or "Clinton Thursday".

The patterns for PERSON, PLACE and THING match similar sets of strings. The pattern for THING accepts acronyms ("I.B.M") and uncapitalized word combinations ("The Lord of the Rings") that would not be accepted by the PERSON pattern. The PLACE pattern attempts to extend matches by appending state and country names ("Waterloo, Ontario, Canada"). The PROPER pattern is a union of the PERSON, PLACE and THING patterns.

When the question analyzer cannot assign a category to a question, the entity extractor generates a set of all 1- to 5-grams within within 30 words of the hotspot. From this set, the entity extractor eliminates n-grams that only appear in a single passage, n-grams that begin or end with prepositions, and n-grams that consist primarily of stopwords and question terms. The remaining n-grams are treated as answer candidates and are passed to the answer-selection component.

2.4 Answer Selection

From the entity extractor, the answer-selection component receives a set of n-grams, a list of the passage and document identifiers where each n-gram is found, and the location of each n-gram within these passages. Along with corpus statistics, this information is used to rank the n-grams. The highest ranking n-gram is returned as the exact answer.

Candidate *redundancy*, the number of distinct passages in which an candidate occurs, has been an important factor in our QA system since TREC-9 [5]. To rank n-gram answer candidates, our TREC 2002 ranking formula combines redundancy with an idf-

table	# elements
Biographies	25,000
Trivia Question and Answers	330,000
Airports(code, names, location)	1,500
Country Locations	800
Country Capitals and Populations	300
Currency by Country	235
Landmark Locations	2,000
Rulers (location,period,title)	25,000
Acronyms	112,000
University and College (name, location)	5,000
Major World Cites (name, location)	21,000
State and Province (name, population, date, capital, bird, flower)	63
Holidays	171
Previous TREC questions and answers	1393
Animal Name (baby, male, female, group)	500

Figure 4: Structured data for early answering.

like weight and information about the location of the candidate relative to the hotspot in passages where it occurs.

The distance of a candidate from a passage hotspot is measured in token positions, with candidates occurring in the hotspot itself treated specially. Given $\mathcal{P}$, an ordered set of m passages, we use the notation $\mathcal{P}_i$ $(1 \leq i \leq m)$ to refer to the ith passage in $\mathcal{P}$. Each passage is split into tokens, where tokens are sequences of alphanumeric characters separated by non-alphanumeric characters.

For each passage P_i containing one or more occurrences of a candidate x $(x \in P_i)$ we determine $loc(P_i, x)$, the distance from the hotspot to the closest occurrence of x If x is contained entirely in the hotspot then $loc(P_i, x) = 0$. Otherwise, $loc(P_i, x) > 0$. For multi-token candidates, the distance is measured from the closest token in the hotspot to the furthest token in the candidate. Thus for candidates occurring before the hotspot, the distance is measured from the start of the candidate to the start of the hotspot, and for candidates occurring after the hotspot the distance is measure from the end of the hotspot to the end of the candidate.

Candidates are then scored using the following formula:

$$\sum_{1 \leq i \leq m,\ x \in P_i} \log \left(\frac{N}{f_x \cdot (loc(P_i, x) + 1)} \right) \qquad (1)$$

where N is sum of the lengths of all documents in the corpus and f_x is the number of occurrences of the candidate in the database.

2.5 Early Answering

The "early answering" subsystem answers questions by referencing a database of structured knowledge gathered from the Web. The subsystem comprises two components: a retrieval component that determines when questions can by answered from the structured knowledge and proposes possible answers, and a justification component that uses IR techniques to identify documents that support the proposed answers.

We gathered Web pages from standard sources like the CIA Factbook and from other sources identified by searching the Web. The Web pages were parsed into tables and manually filtered to remove irregular information. We supplemented the data gathered from the Web with a table of acronyms extracted from the TREC 2001 and 2002 QA corpora and a table of past TREC questions and answers. Figure 4 gives a summary of the tables in the database.

One table in the database contains a collection of trivia questions with their answers. This is the same collection accessed by the passage retrieval component as unstructured text. Here, the trivia collection is treated as structured data and an exact match with the normalized text of a trivia question is required for the associated answer to be proposed. The text of a question is normalized by converting to lower case, removing punctuation and a few stopwords, sorting the words, and eliminating duplicate words.

The early-answering subsystem is geared to answer specific question types. Regular expressions are used to match question forms corresponding to these types

0) *results from the early-answering subsystem*
1) CONTINENT, CURRENCY, LAKE, OCEAN, PLANET, PROVINCE
2) COLOUR, COUNTRY, SEASON, STATE, YEAR
3) ANNIVERSARY, DATE, LENGTH, MOUNTAIN, PERSON, PLACE, PROPER, THING
4) LONG, TIME
5) CODE, LARGE, SPEED
6) NUMBER, RATE, TEMPERATURE
7) MONEY
8) *all other categories*
9) *uncategorized questions*
10) *unanswered questions ("NIL")*

Figure 5: Confidence ranking by category.

(ie. "What is the capital of X?"). Once the question type is identified, the answer is retrieved directly from the corresponding table. When possible, we order the tables so that the first occurrence is the most likely answer. When the time-frame of a question is important, the question is answered in the time-frame of the corpus rather than the time-frame of the gathered data. For example, the question, "Who is the president?" should be answered with "Bill Clinton" rather than "George W. Bush."

To meet the requirements of the QA track, our system not only must return an exact answer, but also must identify a document in the TREC corpus that supports this answer. Given a question and a proposed answer, the justification component searches the TREC corpus for a document containing the answer and keywords from the question in close proximity. If no supporting document can be found, the proposed answer is rejected.

2.6 Merging & Ranking

The final component of our system merges the output of the statistical answer-selection component with the answers generated by the early-answering subsystem and applies a confidence ranking to the result. The decisions made by this merging-and-ranking component are based on our experience with 1393 training questions drawn from the QA tracks for TREC 1999-2001.

Whenever the early-answering subsystem produces an answer for a question in this training set, we judged it to be correct (but not necessarily justified) 96% of the time. Since this performance is superior to that of statistical answer selection for all question categories, answers generated by the early-answering subsystem are always given precedence and are ranked first in the system output. Unfortunately, over the training set, the early-answering subsystem produces an answer for only 12% of the questions.

The ranking of the answers produced by statistical answer selection is entirely based on the question category, with the answers for uncategorized questions ranked lower than categorized questions. The ranking order by category is shown in figure 5. Each line of the figure gives an equivalence class for confidence ranking purposes.

In the rare cases where neither early answering nor statistical answer selection produces an answer, a no-answer ("NIL") result is generated and ranked last. Statistical answer selection fails to produce an answer only when the entity extraction component fails to identify any candidates from the TREC QA corpus. We took this failure as an indication that an answer may not be present in the corpus. This situation is only one in which a "NIL" result is produced by our system.

3 QA Track Results

Our QA track results are summarized in figure 6. The figure reports results for two judgment sets: the official NIST judgments and our own judgments, which were made immediately after the runs were submitted in August. The NIST judgments for uwmtB0 were generated from the answer list provided by NIST, since this run was not officially judged. All the answers we marked correct in this run are exact and supported according to the NIST answer list, but it is possible that some correct answers from this run do not appear in the NIST answer list. In judging our own runs, we were far more forgiving than the NIST judges. Nonetheless, the relative differences between runs under each judgment set is remarkably consistent.

The use of Altavista as a supplement to our own QA resources had an impact of less than 4% on both

	run tag	uwmtB3	uwmtB2	uwmtB1	uwmtB0
	uses early answering?	y	y	n	n
	uses Altavista?	y	n	y	n
NIST judgments	percent correct	36.8%	36.6.X%	33.4%	32.4%
	confidence-weighted score	0.512	0.511	0.441	0.429
MultiText judgments	percent correct	44.4%	44.0%	39.4%	38.6%
	confidence-weighted score	0.614	0.610	0.509	0.493

Figure 6: QA track results.

the number of correctly answer questions and the confidence-weighted score. The use of early answering provided a 10-14% improvement in the number of correctly answered questions and a 16-24% improvement in the confidence-weighted score.

When it was used in a run, the early-answering subsystem generated answers for 65 questions. Of these, 44 (67.7%) were correct. In uwmtB2, where early answering was not used, only 27 of these same 65 questions (41.5%) were answered correctly.

In uwmtB3, our best run, 126 questions could neither be answered by the exact-answering subsystem nor assigned a category from figure 3 by the question analyzer. For these questions we took a purely statistical approach, using n-grams of 1- to 5-words in length as the answer candidates. Of these 126 questions, 27 (21.4%) were answered correctly.

As an additional experiment, we disabled the question categorization and exact answering, and applied purely statistical answer selection to the entire question set. Of the 500 questions, 73 (14.6%) were correct.

4 Web Track Experiments

Our Web track runs were the result of an intensive 48-hour effort intended to resurrect some the older MultiText work in this area and to provide a preliminary evaluation of a few new ideas.

We submitted five runs. One run (uwmtBW0) used only document content for retrieval, one (uwmtBW1) used pagerank in addition to document content, one (uwmtBW2) used anchor text along with document content, one (uwmtBW3) used a combination of document content, pagerank and anchor text, and one (uwmtBW4) used anchor text only.

For content retrieval, we used the current implementation of our *cover-density ranking* algorithm, which is essentially the same technique used in our MultiText TREC-8 Web track runs [6] and is related to the passage-retrieval algorithm used in our QA track runs. The cover-density ranking algorithm locates hotspots that contain combinations of query terms, and bases a document's score on the number and score of the hotspots it contains.

For anchor text retrieval, we applied the cover-density ranking algorithm in a novel way. Anchor points were indexed and hotspots were required to contain an anchor point in addition to the query terms. The score associated with each hotspot was then applied to the score of the document referenced by the anchor, rather than the document where the anchor appears. This approach is similar to other anchor text retrieval techniques [7], but takes advantage of the text surrounding the anchor, as well as the anchor text itself.

Our version of pagerank is a direct implementation of the original pagerank algorithm described by Brin and Page [2]. To apply pagerank to the Web track task, we re-ranked the top 1000 documents retrieved by document content or anchor text according to their pagerank values.

To combine the results of pagerank and/or anchor text retrieval with document content retrieval we simply added the rank of each document in each run and re-sorted them in ascending order. For the purposes of combining runs, a documen. not appearing in a particular run is treated as if it has a rank one greater than the number of documents retrieved by the run.

Our Web track results are summarized in figure 7. In comparison with content-only retrieval, the combination of anchor text and content retrieval provided a 5% improvement in average reciprocal rank and a 7% improvement in precision at 10 documents. Alone, anchor text retrieval exhibited very poor performance. The inclusion of pagerank had a devastating impact on performance. Since our implementation of pagerank has been throughly tested in other projects, we suspect the problem lies in the approach used to incorporate pagerank into our runs.

run tag	uwmtBW0	uwmtBW1	uwmtBW2	uwmtBW3	uwmtBW4
uses document content?	y	y	y	y	n
uses pagerank?	n	y	n	y	y
uses anchor text?	n	n	y	y	n
average reciprocal rank	0.509	0.150	0.535	0.103	0.106
precision at 10	0.747	0.320	0.800	0.260	0.307

Figure 7: Web track results.

5 Conclusions & Future Work

We continue to improve all aspects of our QA system. In future, we intend to expand the number of question categories and improve the recall of the entity extractor on the existing categories. The current version of our early answering system was two-person effort undertaken in the week preceding the question download date; it could be improved with additional answer sources and more a careful answer justification algorithm. In contrast, we expended considerable effort over several months attempting to exploit syntactic information, by parsing retrieved passages and unifying the result with a parse of the question. While we made considerable progress along this path, we ultimately were not able to develop the approach to the point where it made a positive impact on our QA system.

We are encouraged by the results of our Web experiments. In particular, we plan further experiments to explore our approach to using the text surrounding anchors. Our use of pagerank was a failure. In future, we hope to determine how to exploit pagerank more effectively in our system.

References

[1] Steven Abney. Partial parsing via finite-state cascades. *Journal of Natural Language Engineering*, 2(4):337–344, December 1996.

[2] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. In *Seventh International World Wide Web Conference*, pages 107–117, Brisbane, Australia, April 1998.

[3] C. L. A. Clarke, G. V. Cormack, T. R. Lynam, C. M. Li, and G. L. McLearn. Web reinforced question answering. In *2001 Text REtrieval Conference*, Gaithersburg, MD, 2000.

[4] Charles L. A. Clarke, G. V. Cormack, and F. J. Burkowski. An algebra for structured text search and a framework for its implementation. *Computer Journal*, 38(1):43–56, 1995.

[5] Charles L. A. Clarke, Gordon V. Cormack, and Thomas R. Lynam. Exploiting redundancy in question answering. In *24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 358–365, New Orleans, September 2001.

[6] G. V. Cormack, C. L. A. Clarke, C. R. Palmer, and D. I. E. Kisman. Fast automatic passage ranking. In *Eighth Text REtrieval Conference (TREC-8)*, Gaithersburg, Maryland, November 1999.

[7] Nick Craswell, David Hawking, and Stephen Robertson. Effective site finding using link anchor information. In *24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 250–257, New Orleans, September 2001.

[8] Susan Dumais, Michele Banko, Eric Brill, Jimmy Lin, and Andrew Ng. Web question answering: Is more always better? In *25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Tampere, Finland, August 2002.

[9] Jimmy Lin. The Web as a resource for question answering: Perspectives and challenges. In *3rd International Conference on Language Resources and Evaluation*, May 2002.

Acknowledgments

Our thanks to Ed Fox and Virginia Tech for providing us with the computing resources and network bandwidth to generate the TB Web crawl used in our QA track runs.

The YorkQA Prototype Question Answering System

Marco De Boni, Jose-Luis Jara-Valencia, Suresh Manandhar

Department of Computer Science,
University of York,
York Y010 5DD

DRAFT VERSION - DUE FOR REVISION

Abstract

A preliminary analysis of our QA system implemented for TREC-11 is presented, with an initial evaluation.

Introduction

The aim of our system in this year's TREC QA track was 1) to see how much our previous system could be improved simply by removing bugs and inconsistencies and 2) to test new techniques on "real" data.

The system produced was therefore again a prototype experimental system rather than a "complete" machine ready for deployment. In particular it still lacks an information retrieval engine (we relied again on the documents retrieved by NIST's PRISE IR engine), and has only an outline Named Entity Recogniser and an incomplete answer extraction module. Nevertheless it did give an indication as to which ideas are most promising and should be investigated further, in particular as regards determining an answer in a sentence.

Improvements on the previous version

A close examination of the system used for TREC 2001 [Alfonseca et al. 2002] revealed a number of small bugs across all modules which were significant enough to affect performance. These were corrected for this year's entry. Furthermore, a close analysis of our results taking into account the contribution of each module revealed that a number of components we used did not improve performance and in fact were detrimental. In particular, the Noun Phrase Chunker which we used last year was removed as its output was not precise enough to be used productively.

The question recogniser which we previously used proved very hard to maintain and improve, based as it was on a large number of patterns and exceptions with very limited use of linguistic resources. It was therefore re-written from scratch in a much more elegant way in order to make much more use of linguistic resources such as WordNet.

The Named Entity recogniser was also rewritten from scratch, as was the answer extractor, which had to cope with this year's track aim, which was to extract an exact answer as opposed to a string of words.

Question Recogniser

A simple set of rules allowed the question recogniser to determine the focus of the question. The initial recognition, however, was too refined to be of much use: given a question such as"What president's wife commented on the affair", it would return a question type of "president's wife". The WordNet is-a hierarchy was therefore used to determine a less fine-grained answer type.

WordNet, however, is deceptive without accurate word-sense disambiguation: satellite is a person, if satellite is intended in the sense of synset 107546753, "a person who follows or serves another", but not if it is intended in the more common meaning of synset 103275905, "a man-made object that orbits around the earth". Given that we did not have an accurate word-sense disambiguation module, we resorted to assuming that the meaning of any word was the most common word and hand-crafted a series of rules which reflected this (e.g. our rules state that "satellite" is not a person, but is a "thing").

On the other hand WordNet could not be used to determine whether a question was looking for MUC entities such as locations due to inconsistencies such as the fact that according to the hypernym hierarchy, a city is a location, but a lake is not; again hand-crafter rules were applied to state facts such as "sea is a location" and hence determine that a question asking “What sea…” could be answered by looking for a location Named Entity.

Named Entity Recogniser

Texts were initially tokenized by applying several hand-coded heuristics. Then they were tagged using the TnT part-of-speech tagger (Brants 2000). Sentences were then split by an algorithm that utilises a number of heuristics and a list of abbreviations extracted by another algorithm that uses active learning when the case is highly ambiguous. There is no formal evaluation of this sentence splitter, but it appears to work well in most of cases.

For the Named entity recognition YorkQA system uses our own implementation of Nymble (Bikel et.al. 1997) which utilises hidden Markov models to identify named entities in the text. An improved version of this algorithm showed high accuracy in MUC-7. We estimates that the this version of Nymble is reaching about 70% recall and 80% precision.

These results are a big improvement form YorkQA-2001. However, the program identifies only MUC entities (person names, organisation names, location names, dates, time expressions, currency expression and percentage expression). This means that for question about other type of entities (speeds, distances, durations, etc.) our system has much less information to work with.

We also detected that in question asking for fine-grain pieces of information (e.g. the first name of a person), this module turned out to be inappropriate.

Semantic Distance Metrics

The semantic distance metric used for the TREC-10 was improved (see De Boni and Manandhar 2002 for an in-depth explanation of the implementation) and then subjected to a thorough analysis of its components. In particular we evaluated different approaches to the measure of semantic relevance for question answering in order to decide how the use of WordNet information, part-of-speech information and head-phrase chunking could provide a reliable measurement for semantic relevance. It was seen that a semantic distance metric based on all WordNet relations (is-a, satellite, similar, pertains, meronym, entails, etc) and using compound word information, proper noun identification, part-of-speech tagging and word frequency information gave best results.

Answer Word Finder

Our Named Entity recogniser only recognised a small number of entity types, which did not correspond to all types identified by the question type recogniser. In order to determine possible answers, the system made use of Wordnet's is-a relationships, looking in the answer sentence which had the highest semantic similarity measure for hypernyms of the question type. This approach appeared to be fruitful in determining "rare" entities which it would be impractical to build a Named Entitiy recogniser to tag: a good example was the question asking the name of King Arthur's sword: the question type was "sword", the answer found was Excalibur as Excalibur was listed in WordNet as a hypernym of sword.

Conclusions and Further Work

This year's task confirmed that Question Answering is a complex task which requires accurate component parts: underperformance in any of the components results in a deterimental effect to the system as a whole. In such a complex system, however, it is not a simple task to determine why things go wrong and indeed why things go right. Nevertheless is is important to determine how the individual parts interact in order to improve performance, and, while such an analysis can be extremely laborious, it is necessary. The disadvantages of working in a very small team were also highlighted, as the complexity of the problem requires extensive work in disparate area.

Future work will include an extension an improvement of our Named Entity component, the use of an information retrieval engine (as opposed to relying on the documents provided by NIST), improvements in the answer locator and possibly some inference mechanism.

Bibliography

Alfonseca, E., De Boni, M., Jara, J.L., Manandhar, S., "A prototype Question Answering system using syntactic and semantic information for answer retrieval", in Proceedings of the 10th Text Retrieval Conference (TREC-10), Gaithersburg, US, 2002.

Bikel, D., Miller, S, Schwartz, R. and Weischedel, R. "Nymble: a high-performance learning name-finder". In Proceedings of the Fifth Conference on Applied Natural
Language Processing, 194--201, 1997.

Brants, T., "TnT - A statistical Part-Of-Speech tagger", User Manual, 2000.

De Boni, M., and Manandhar, S., "Automated discovery of telic relations for WordNet", in Proceedings of the first International WordNet conference, India, 2002.

Web Document Retrieval Using Sentence-query Similarity

Eui-Kyu Park, Seong-In Moon and Dong-Yul Ra
Computer Science Dept., Yonsei University
{ ekpark, simoon, dyra, }@dragon.yonsei.ac.kr

Myung-Gil Jang
Electronics and Telecommunications Research Institute
mgjang@etri.re.kr

1. Introduction

For the web document retrieval experiments in our TREC '2002 participation, we used two new methods. One is the use of anchor texts, which has been advocated by many researchers. But the methods used by them is different from our method. The second is the use of sentence-query similarity.

It has been known that the use of links for web retrieval did not show impressive improvement in performance [5,6,8,9]. But Bailey, etc. [1] reported that using anchor texts can improve retrieval performance. However, our home page finding experiment done for TREC '2001 showed that it is not the case. The use of anchor texts did not allow any improvement in performance. Our method to use the anchor texts this year is changed a lot from last year and found that it is pretty effective.

The major focus of our experiment this year is in the use of sentential information in information retrieval. We obtain similarity values between sentences of a document and the query and use them for computing the retrieval score of the document. The main idea is the following: a sentence in a document that is much relevant to the query can support relevance of the document to the query. We compute the similarity between each sentence in the document and the query. The degree of this similarity is incorporated in calculating the document's score (in addition to the similarity between the document as a whole and the query). It has been found that it does not take too much time for this extra processing. Our experiment showed that including the sentential information in the proposed way can significantly improve retrieval effectiveness.

2. Use of sentence-query similarity

2.1 Motivation

Let us start by looking at an example. Assume that the query is "the museums in Philadelphia." Let us consider the two documents D_i and D_j as shown below.

> D_i : The *museum* of natural history in Chicago is famous. Its huge size surprised a student from *Philadelphia* who was traveling with his family. .....

> D_j : John visited a *museum* located in *Philadelphia* after he looked around the University of Pennsylvania campus. The museum contained a lot of things that reveals the nature of American culture. ......

The set of index terms of the query is {museum, Philadelphia}. Taking the document as a whole to match against the query, the relevance of D_i looks almost same as that of D_j since both documents have all of the index terms in the query.

Note that the query terms are distributed throughout the sentences in D_i, but all the query terms appear in a same sentence in D_j (the first one in this case). We argue that having most of the query terms in the same sentence strongly indicates that the document is relevant to the query. This argument works in this example, i.e., D_j is more relevant to the query than D_i.

The ideal way of retrieving documents for a query would be to use the meanings of the sentences in the documents. This indicates that the similarity between a sentence and the query have to play an important role. We try to find out how similar each sentence in the document is to the query. This result must be involved in determining the retrieval score of the document.

The best way to compare a sentence with a query would be to compare their meanings. But the state of the art of natural language processing does not allow this. There does not exist any system yet that can interpret meanings of arbitrary sentences stably. Therefore, it is not possible to build a practical information retrieval system that compares the meanings in computing similarity between a sentence and a query. However, we still want to use sentence-query similarity for information retrieval.

The method that is adopted should be a one that can lead to the practical systems. We decided to use a simple measure for similarity, i.e., the number of common words between the sentence and the query. (This measure is very crude now. But it seems to work and can be replaced by a better one if it is found later.)

2.2. Similarity computation

We adopt the vector-space model to compute the document-query similarity $sim(D,Q)$. Cosine coefficient is used to measure this similarity. Thus the retrieval relevance score of a document D is

$$RSV(D,Q) = sim(D,Q) \qquad (1)$$

To include the sentence-query similarity in the relevance score of D, the next formula is used.

$$RSV(D,Q) = sim(D,Q) + \alpha \sum_{i=1}^{n} C(S_i,Q) \qquad (2)$$

The second term on the right-hand side is the contribution by the sentence-query similarity. (The number of sentences in the document is denoted by n.) $C(S_i,Q)$ denotes the similarity between S_i (the i^{th} sentence in D) and the query Q. Instead of using sophisticated techniques such as natural language processing, computing $C(S,Q)$ is based on the degree of co-occurrence of words between S and Q. It is computed as

$$C(S,Q) = \begin{cases} \left(\dfrac{|S \cap Q|}{|Q|} \right)^k & \text{if } |S \cap Q| \geq \tau(|Q|) \\ 0 & \text{otherwise} \end{cases} \qquad (3)$$

$|S \cap Q|$ represents the count of the common indexing terms between S and Q. $|Q|$ denotes the number of indexing terms in Q.

The constant α in equation (2) works as a weighting factor for the contribution by the sentence-query similarity. The exponent k in equation (3) is used to control the degree of importance of the high values of the ratio $|S \cap Q| / |Q|$ compared to the lower ones. As k increases, the high ratio becomes more important than the lower ones. $\tau(|Q|)$ is used to nullify the sentential contribution in the cases where the number of common words is small. Currently $\tau(1) = 2$, $\tau(2)=1$, $\tau(3)=2$, $\tau(4)=2$, $\tau(5)=2$, and $\tau(i)=3$ for $i \geq 6$.

3. Using anchor texts

Even though the use of anchor texts did not result in any noticeable performance improvement last year we decided to continue to use anchor texts. But we used it in a different way this year. Let's assume that document D is pointed to by links with anchor texts L_i, $i=1 \ldots l$. Let $D_{a(i)}$ denote the document which contains the anchor text L_i.

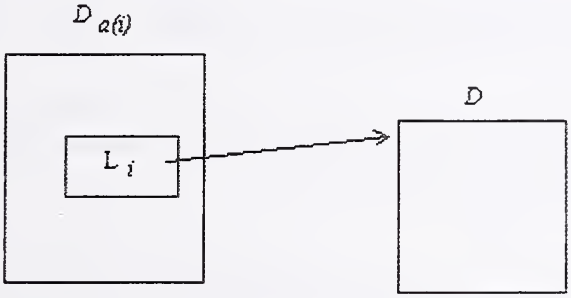


Fig.1: Using anchor texts of incoming links

For a document its incoming links' anchor texts takes part in computing its relevance. In Fig. 1 the anchor text L_i is involved in computing the relevance of D.[1] But outgoing link's anchor text is not used in this processing. Thus the anchor text L_i does not give any contribution to $D_{a(i)}$ as far as links are concerned. We have two methods for utilizing anchor texts.

- Method1 (AT): This one is what was used last year. It uses the cosine coefficient measure to compute the similarity between the anchor text and the query. The contribution by the anchor texts is computed as

$$\sum_{i=1}^{l} sim(L_i, Q) \qquad (4)$$

where *sim* represents the cosine coefficient measure.

[1] An anchor text is considered to be a part of the text of the document containing it. Thus L_i is also used in computing the relevance of $D_{a(i)}$ based upon the vector-space model.

- Method2 (BETA): We additionally use this second method this year. In this method the way we use anchor texts is similar to the use of sentences in the previous section. The incoming link's anchor text L_i in $D_{a(i)}$ is treated like a sentence in D. But the weight given to the similarity between an anchor text and the query can be different from that between a sentence and a query. The importance of the anchor text for the relevance of D can be different from that of a sentence in D. The contribution by the anchor texts L_i's whose links point to document D to the relevance of D is computed as

$$\beta \sum_i C(L_i, Q) \tag{5}$$

The constant β is used as the weighting factor for the anchor text-query similarity. The same similarity measure C is used.

4. The named page finding task

For the named page finding task the relevance score is obtained by incorporating all contributions discussed above.

$$RSV(D,Q) = sim(D,Q) + \alpha \sum_i C(S_i,Q) + \sum_{i=1}^{l} sim(L_i,Q) + \beta \sum_{i=1}^{l} C(L_i,Q) \tag{6}$$

We show the results of experiments related to this task ARR stands for the average(mean) reciprocal rank which is used for indicating performance of systems for this type of tasks.

- Official run :

We submitted one official run on this task whose evaluation is given below. Total number of topics is 150. Column 2 shows the average reciprocal rank(ARR). The third column displays the number of topics for which the answer exists among the top 10 documents of the ranked list. The fourth column is the number of topics for which the answer does not exist in the top 50 of the ranked list. The second row is data from the official run. The third row is about the run with better performance which was obtained at a later experiment with more tuning.

Table 1: Performance in named page finding task

Run	ARR	# topics found in top 10	# topics not found
Official	0.671	124 (82.7%)	13 (8.7%)
Best	0.697	128 (85.3%)	14 (9.3%)

$\alpha = 2$ $\beta = 10$ $k=3$

- Experimentation on the effects of α, β, and k :

Our relevance computation is dependent on the constants α, β, and k. We performed some experiments to find out the best combination of the values of these constants. Fig. 2 shows the result of this experiment ("A" in the figure denotes the parameter α). According to the result, setting α to 1 is recommended. For this best α value, the system performs best when β = 4.

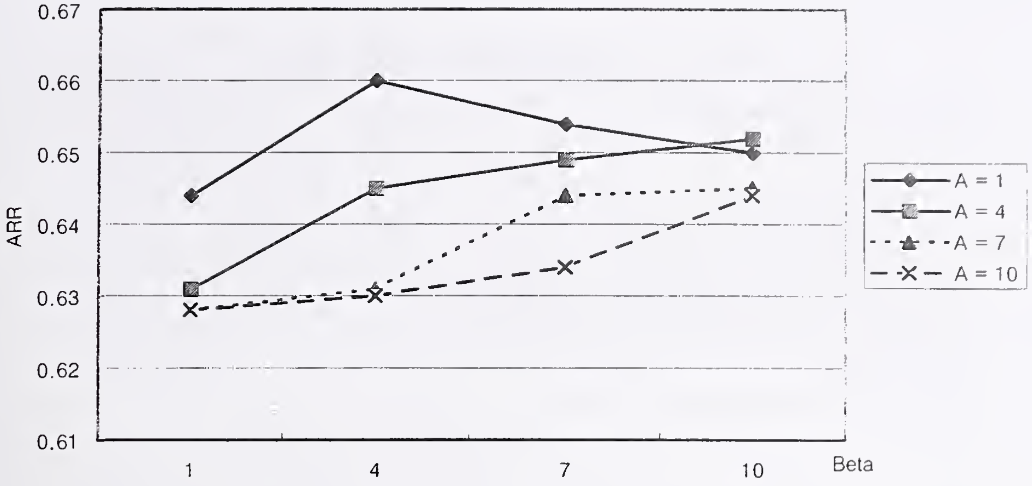


Fig 2: ARR plots for various α, β (with fixed k = 3)

- The effect of k can be shown in Fig.3 . The best performance is obtained when k = 5.

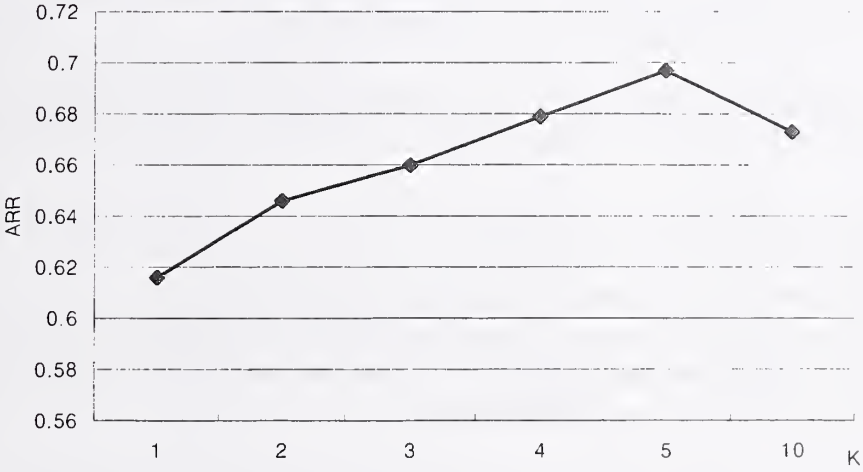


Fig 3: ARR plotted for various k (α=1, β=4)

- Experimentation on the effects of information sources of retrieval

There are several sources that contribute to the relevance of a document. They are document-query similarity obtained by the vector space model (VS), sentence-query similarity (S), similarity between the anchor texts and query by cosine(AT), and anchor text-query similarity obtained by counting common words(BETA). The runs with some of these were generated. CUT in the last row in Table 2 means that the documents are removed from the ranked list when they do not receive positive values from either S or BETA.

Table 2: Various information sources' effects

Runs	ARR	top10	% of top 10	NF	% of NF
np_VS.txt	0.567	118	78.7	16	10.7
np_VS_S.txt	0.641	122	81.3	17	11.3
np_VS_S_AT.txt	0.667	126	84.0	17	11.3
np_VS_S_AT_BETA.txt	0.695	126	84.0	15	10.0
np_VS_S_AT_BETA_CUT.txt	0.697	128	85.3	14	9.3

(α =1, β=4, k=5
NF stands for "not found in top 50".)

This table shows that using sentence-query similarity enables the system to achieve a significant increase in performance. We also observed a noticeable improvement in performance by the use of anchor texts, which were not seen last year.

5. Topic distillation task

In this task we need to find the key resources. They are the documents from which the low and more specific ones can be reached. We want to return a few key resources rather than many low quality or peripheral documents. To make the key resources go up in the ranked list we use the following heuristic:

> For any two documents D_i and D_j in the relevant list (the result of search by a metric such as Eq. 6), increase score of D_i by the amount of score of D_j if D_j is pointed to by a link that exists in D_i.

Therefore, the relevant documents which have many relevant children will get the increased score. In the current implementation only the immediate child can increase its parent's score. The run submitted is shown in Table 3. The result illustrates that our system is not good at the topic distillation task.

One of the reason for the poor performance is that the concept of topic distillation is not clear. It can be either a home page or a specific web page while it can be a sub site. We could not come up with a technique that can identify key resources since our understanding on this concept is obscure.

6. Summary

In computing the retrieval status value of a document sentence-query similarity is incorporated. In addition the anchor text of incoming links is used in a similar way. The requirement for building practical systems made us use a simple scheme in computing this similarity, which is actually the word co-occurrence count. It has been observed that incorporating sentence-query similarity leads to significant increase in performance in the named page finding task. Using anchor texts in a similar way also leads to a better system. For the topic distillation task a simple heuristic has been used but the experimental result showed that it did not worked well.

7. References

[1] P. Bailey, N. Craswell and D. Hawking, “Engineering a multi-purpose test collection for Web Retrieval experiments,” *Information Processing and Management*, In press.

[2] S. Fujita, "Reflections on "Aboutness" TREC-9 Evaluation Experiments at Justsystem," In Proceedings of the Ninth Text Retrieval Conference (TREC-9), National Institute for Standards and Technology, 2000.

[3] J.M. Kleinberg, “Authoritative sources in a hyperlinked environment,” In Proceedings of 9th ACM-SIAM Symposium on Discrete Algorithms, p. 668-677, 1998.

[4] J-M Lim, H-J Oh, S-H Myaeng and M-H Lee, “Improving Efficiency with Document Category Information in Link-based Retrieval,” in Proceedings of the Information Retrieval on Asian Languages Conference, 1999.

[5] W. Kraij and T. Westervel, “TNO/UT at TREC-9: How different are Web documents?” In Proceedings of the Ninth Text Retrieval Conference (TREC-9), National Institute for Standards and Technology, 2000.

[6] J. Savoy and Y. Rasolofo, “Report on the TREC-9 Experiment: Link-Based Retrieval and Distributed Collections,” In Proceedings of the Ninth Text Retrieval Conference (TREC-9), National Institute for Standards and Technology, 2000.

[7] A. Singhal and M. Kaszkiel, “AT&T at TREC-9,” In Proceedings of the Ninth Text Retrieval Conference (TREC-9), National Institute for Standards and Technology, 2000.

[8] D. Hawking, “Overview of the TREC-9 Web Track,” In Proceedings of the Ninth Text Retrieval Conference (TREC-9), National Institute for Standards and Technology, 2000.

[9] D. Hawking, E. Voorhees, N. Craswell and P. Bailey, "Overview of the TREC-8 Web Track," In Proceedings of the Ninth Text Retrieval Conference (TREC-8), National Institute for Standards and Technology, 1999.

Table 3: Performance on topic distillation task

Run id: yedi01	Run description: automatic, title only, link(anchor text)		No. of topics: 50
Total number of documents over all topics			
Retrieved: 31072	Relevant: 1574	Relevants retrieved: 640	
Recall level precision averages		Document level precision averages	
Recall	Precision	Recall	Precision
0.0	0.3886	At 5 docs	0.1755
0.1	0.2797	At 10 docs	0.1510
0.2	0.1945	At 15 docs	0.1361
0.3	0.1223	At 20 docs	0.1255
0.4	0.1060	At 30 docs	0.1136
0.5	0.0753	At 100 docs	0.0639
0.6	0.0411	At 200 docs	0.0424
0.7	0.0282	At 500 docs	0.0215
0.8	0.0207	At 1000 docs	0.0131
0.9	0.0081		
1.0	0.0045		
Average precision (non-interpolated) : 0.0986		R-precision (exact) : 0.1298	

TREC 2002 Results

APPENDIX A

This appendix contains the evaluation results for TREC 2002 runs. The initial pages list each of the runs (identified by the run tags) that were included in the different tracks. Associated with each tag is the organization that produced the run and additional information such as whether the queries were produced manually or automatically as appropriate. Following the run list is a description of the evaluation measures used in most tracks. For more details about the measures used in a particular track, see the overview paper for that track. The remainder of the appendix contains the evaluation results themselves, in the order given in the run list.

Cross-Language Track

Tag	Organization	Topic Language	Run Type
BBN11XLA	BBN	English	Automatic, title+desc
BBN11XLB	BBN	English	Automatic, title+desc
BBN11XLC	BBN	English	Automatic, title+desc
BBN11XLS	BBN	English	Automatic, title+desc
humAR02td	Hummingbird	Arabic	Automatic, title+desc
humAR02tde	Hummingbird	Arabic	Automatic, title+desc
humAR02tdm	Hummingbird	Arabic	Automatic, title+desc
humAR02tdne	Hummingbird	Arabic	Automatic, title+desc+narr
humAR02te	Hummingbird	Arabic	Automatic, title
ibmy02d	IBM T.J. Watson Res. Ctr.-Ittycheriah	Arabic	Automatic, title
ibmy02a	IBM T.J. Watson Res. Ctr.-Ittycheriah	English	Automatic, title+desc
ibmy02b	IBM T.J. Watson Res. Ctr.-Ittycheriah	English	Automatic, title+desc
ibmy02c	IBM T.J. Watson Res. Ctr.-Ittycheriah	English	Automatic, title+desc
iit02mel	IIT Info. Retrieval Lab	Arabic	Automatic, title+desc
iit02mnl	IIT Info. Retrieval Lab	Arabic	Automatic, title+desc
iit02mpl	IIT Info. Retrieval Lab	Arabic	Automatic, title+desc
iit02xma	IIT Info. Retrieval Lab	English	Automatic, title+desc
iit02xst	IIT Info. Retrieval Lab	English	Automatic, title+desc
apl11cal	Johns Hopkins Univ.	Arabic	Automatic, title+desc
apl11ce1	Johns Hopkins Univ.	English	Automatic, title+desc
apl11ce2	Johns Hopkins Univ.	English	Automatic, title+desc
apl11ce3	Johns Hopkins Univ.	English	Automatic, title+desc
apl11ce4	Johns Hopkins Univ.	English	Automatic, title+desc
BKYMON	Univ. of Calif./Berkeley	Arabic	Automatic, title+desc
BKYCL1	Univ. of Calif./Berkeley	English	Automatic, title+desc
BKYCL2	Univ. of Calif./Berkeley	English	Automatic, title+desc
BKYCL3	Univ. of Calif./Berkeley	English	Automatic, title+desc
AutoMono	Univ. of Md./Oard	Arabic	Automatic, title+desc
ManMono	Univ. of Md./Oard	Arabic	Manual
AutoClirDoc	Univ. of Md./Oard	English	Automatic, title+desc
AutoClirExp	Univ. of Md./Oard	English	Automatic, title+desc
UMassM	Univ. of Massachusetts	Arabic	Automatic, title+desc
UMassX2	Univ. of Massachusetts	English	Automatic, title+desc
UMassX2n	Univ. of Massachusetts	English	Automatic, title+desc+narr
UMassX6	Univ. of Massachusetts	English	Automatic, title+desc
UMassX6n	Univ. of Massachusetts	English	Automatic, title+desc+narr
UniNE1	Univ. of Neuchatel	Arabic	Automatic, title+desc
UniNE2	Univ. of Neuchatel	Arabic	Automatic, title
UniNE3	Univ. of Neuchatel	Arabic	Automatic, title+desc+narr
UniNE4	Univ. of Neuchatel	Arabic	Automatic, title+desc

Filtering Track, Adaptive Task

Tag	Organization	Resources Used	Optimized For
CMUDIRFDESC	Carnegie Mellon Univ.	None	T11F
CMUDIRFml	Carnegie Mellon Univ.	None	T11F
CMUDIRUDESC	Carnegie Mellon Univ.	None	T11U
CMUDIRUml	Carnegie Mellon Univ.	None	T11U
ICTAdaFT11Ua	Chinese Academy of Sciences	None	T11U
ICTAdaFt11Ub	Chinese Academy of Sciences	None	T11U
ICATAdaFt11Uc	Chinese Academy of Sciences	None	T11U
ICATAdaFt11Ud	Chinese Academy of Sciences	None	T11U
reliefst11u	CLIPS-IMAG Lab	None	T11U
FDUT11AF1	Fudan Univ.	None	T11U
FDUT11AF2	Fudan Univ.	None	T11F
dimacsdd102a	Consultant-Lewis	Other TREC	T11U
dimacsdd102b	Consultant-Lewis	Other TREC	T11F
Iritsigal	IRIT/SIG	Other TREC	T11U
iritsiga2	IRIT/SIG	Other TREC	T11U
apl11Fah1	Johns Hopkins Univ.	Other TREC	T11U
apl11Fah2	Johns Hopkins Univ.	Other TREC	T11U
apl11Faq1	Johns Hopkins Univ.	Other TREC	T11U
apl11Faq2	Johns Hopkins Univ.	Other TREC	T11U
KerMITT11af1	KerMIT Consortium	None	T11U
KerMITT11af2	KerMIT Consortium	None	T11U
KerMITT11af3	KerMIT Consortium	None	T11F
KerMITT11af4	KerMIT Consortium	None	T11U
ok11aflb	Microsoft Research Cambridge	None	T11F
ok11aflu	Microsoft Research Cambridge	None	T11U
ok11afsb	Microsoft Research Cambridge	None	T11F
ok11afsu	Microsoft Research Cambridge	None	T11U
pirc2F01	Queens College, CUNY	Other TREC	T11U
pirc2F02	Queens College, CUNY	Other TREC	T11U
pirc2F03	Queens College, CUNY	Other TREC	T11U
pirc2F04	Queens College, CUNY	Other TREC	T11U
dimacs11a30Q	Rutgers Univ.-Kantor	Other TREC	T11U
dimacs11aAPQ	Rutgers Univ.-Kantor	Other TREC	T11U
dimacs11aP1Q	Rutgers Univ.-Kantor	Other TREC	T11U
thuT11af1	Tsinghua Univ.	Other Reuters, other TREC	T11F
thuT11af2	Tsinghua Univ.	Other Reuters	T11F
thuT11af3	Tsinghua Univ.	Other Reuters	T11F
cedar02affb0	Univ. of Buffalo-Cedar	None	T11F
cedar02afut0	Univ. of Buffalo-Cedar	None	T11U

Filtering Track, Batch Task

Tag	Organization	Resources Used	Optimized For
ICTBatFT11Ua	Chinese Academy of Sciences	None	T11U
ICTBatFT11Ub	Chinese Academy of Sciences	None	T11U
CCT11BFC	Clairvoyance Corp.	Other data	Neither
CCT11BFD	Clairvoyance Corp.	Other Data	Neither
iritsigb	IRIT/SIG	Other TREC	T11U
apl11FbF	Johns Hopkins Univ.	Other TREC	T11F
apl11FbSU	Johns Hopkins Univ.	Other TREC	T11U
KerMITT11bf2	KerMIT Consortium	None	T11U
mma2002003	Moscow Medical Academy	Other data	T11U
kNII11bf1	NII/RCIR	None	Neither
kNII11bf2	NII/RCIR	None	Neither
dimacs11b001	Rutgers Univ.-Kantor	None	T11U
cedar02bffb0	Univ. of Buffalo-Cedar	None	T11F
cedar02bfut0	Univ. of Buffalo-Cedar	None	T11U
UNTextCatF	Univ. of North Texas	None	T11F
UNTextCatSU	Univ. of North Texas	None	T11U

Filtering Track, Routing Task

Tag	Organization	Resources Used	Optimized For
ICTRouFT11Ua	Chinese Academy of Sciences	None	T11U
ICTRouFT11Ub	Chinese Academy of Sciences	None	T11U
iritsigr	IRIT/SIG	Other TREC	Neither
apl11Frm	Johns Hopkins Univ.	Other TREC	Neither
apl11Frsvm	Johns Hopkins Univ.	Other TREC	Neither
KerMITT11rr2	KerMIT Consortium	None	T11U
msPUMb	Microsoft Research Cambridge	Other TREC	Neither
msPUMs	Microsoft Research Cambridge	Other TREC	Neither
mma2002001	Moscow Medical Academy	Other data	T11U
mma2002002	Moscow Medical Academy	Other data	T11U
QUT	Queensland Univ. of Tech.	None	T11U
UHcl1	Univ. of Hertfordshire	Other data	Neither
UHcl2	Univ. of Hertfordshire	Other data	Neither
UNTextCatR	Univ. of North Texas	None	Neither
UNTextCatR1	Univ. of North Texas	None	Neither

Novelty Track

Tag	Organization
cmu02t300rAs	Carnegie Mellon University (CMUDIR)
cmu02t300rBw	Carnegie Mellon University (CMUDIR)
cmu02t300rCb	Carnegie Mellon University (CMUDIR)
cmu02t300rCv	Carnegie Mellon University (CMUDIR)
cmu02t300rCw	Carnegie Mellon University (CMUDIR)
novcolcl35	Columbia University-Schiffman
novcolcl85	Columbia University-Schiffman
novcolclfx	Columbia University-Schiffman
novcolmerg	Columbia University-Schiffman
novcolsent	Columbia University-Schiffman
fdut11n1	Fudan University
fdut11n2	Fudan University
fdut11n3	Fudan University
dumbrun	Institut de Recherche en Informatique de Toulouse (IRIT/SIG)
ntu1	National Taiwan University
ntu2	National Taiwan University
ntu3	National Taiwan University
nttcslabnvp	NTT Communication Science Laboratories
nttcslabnvr2	NTT Communication Science Laboratories
pircs2N01	Queens College, CUNY
pircs2N02	Queens College, CUNY
pircs2N03	Queens College, CUNY
pircs2N04	Queens College, CUNY
pircs2N05	Queens College, CUNY
ss1	StreamSage, Inc.
thunv1	Tsinghua University
thunv2	Tsinghua University
thunv3	Tsinghua University
thunv4	Tsinghua University
thunv5	Tsinghua University
UAmsT11ntcom	University of Amsterdam-Monz
UAmsT11ntlem	University of Amsterdam-Monz
UAmsT11ntste	University of Amsterdam-Monz
UIowa02Nov4	University of Iowa
UIowa02Nov5	University of Iowa
CIIR02tfkl	University of Massachusetts
CIIR02tfnew	University of Massachusetts
UMICH4	University of Michigan
UMIch2	University of Michigan
UMich3	University of Michigan
UMich5	University of Michigan
Umich1	University of Michigan

Question Answering Track, List

Tag	Organization
clr0211	CL Research
clr0212	CL Research
clr0213	CL Research
LCClist2002	Language Computer Corp.
SUT11IR1LT	Syracuse Univ.
SUT11IR1LT2	Syracuse Univ.
UdeMlistNoW	Univ. of Montreal
sheft11lo	Univ. of Sheffield, Western Bank
sheft11log	Univ. of Sheffield, Western Bank

Question Answering Track, Main

Tag	Organization
ali2002b	Alicante University
BBN2002A	BBN
BBN2002B	BBN
BBN2002C	BBN
CMUJAV000495	Carnegie Mellon Univ.-Javelin
CMUJAV000501	Carnegie Mellon Univ.-Javelin
ICTQA11a	Chinese Academy of Sciences
ICTQA11b	Chinese Academy of Sciences
ICTQA11c	Chinese Academy of Sciences
clr02b1	CL Research
clr02b2	CL Research
2002cuaq1	Columbia Univ.-Illouz
2002cuaq2	Columbia Univ.-Illouz
FDUT11QA1	Fudan University
ibmsqa02a	IBM T.J. Watson Res. Ctr.-Ittycheriah
ibmsqa02b	IBM T.J. Watson Res. Ctr.-Ittycheriah
ibmsqa02c	IBM T.J. Watson Res. Ctr.-Ittycheriah
IBMPQ	IBM T.J. Watson Res. Ctr.-Prager
IBMPQSQA	IBM T.J. Watson Res. Ctr.-Prager
IBMPQSQACYC	IBM T.J. Watson Res. Ctr.-Prager
exactanswer	InsightSoft-M
IRST02D1	ITC-irst
IRST02D2	ITC-irst
IRST02D3	ITC-irst
LCCmain2002	Language Computer Corporation
limsiQalir1	LIMSI
limsiQalir2	LIMSI
limsiQalir3	LIMSI
aranea02a	MIT
aranea02pbq	MIT
aranea02pc3	MIT
nuslamp2002	National Univ. of Singapore-Lee
pris2002	National Univ. of Singapore-Hui
Nttcs11qam	NTT Communication Science Labs
KLE000	POSTECH
SUT11IR1MT	Syracuse University
MITRE2002B	The MITRE Corp.
TUSRUN1	Tokyo University of Science
UAmsT11qaM1	Univ. of Amsterdam-Monz
UAmsT11qaM2	Univ. of Amsterdam-Monz
UAmsT11qaM3	Univ. of Amsterdam-Monz
leria2002	Universit d'Angers
LIA2002a	University of Avignon
UIUC2002	Univ. of Illinois at Urbana/Champaign
UIowaQA0201	Univ. of Iowa
UIowaQA0202	Univ. of Iowa
UIowaQA0203	Univ. of Iowa
DLT02QA01	Univ. of Limerick
DLT02QA02	Univ. of Limerick
NSIRG	Univ. of Michigan

Question Answering Track, Main (continued)

Tag	Organization
NSIRGN	Univ. of Michigan
NSIRN	Univ. of Michigan
UdeMmainNoW	Univ. of Montreal
UdeMmainWeb	Univ. of Montreal
pqas21	Univ. of Pisa
pqas22	Univ. of Pisa
pqas23	Univ. of Pisa
sheft11mo3	Univ. of Sheffield, Western Bank
sheft11mog1	Univ. of Sheffield, Western Bank
sheft11mog3	Univ. of Sheffield, Western Bank
ilv02t	Univ. of Southern California/ISI
ilv02wt	Univ. of Southern California/ISI
isi02	Univ. of Southern California/ISI
uwmtB1	Univ. of Waterloo
uwmtB2	Univ. of Waterloo
uwmtB3	Univ. of Waterloo
yorkqa01	Univ. of York

Web Track, Named Page Finding Task

Tag	Organization	Document Structure	Anchor Text	Link Structure
ajouai0201	Ajou Univ.	Yes	No	No
ajouai0202	Ajou Univ.	Yes	No	No
ajouai0203	Ajou Univ.	Yes	No	No
ajouai0204	Ajou Univ.	Yes	No	No
ajouai0205	Ajou Univ.	Yes	No	Yes
LmrAllEq	Carnegie Mellon University	Yes	Yes	No
LmrAllEst	Carnegie Mellon University	Yes	Yes	No
LmrDocStruct	Carnegie Mellon University	Yes	No	No
LmrNoStruct	Carnegie Mellon University	No	Yes	No
LmrSmall	Carnegie Mellon University	Yes	Yes	No
ictnp2	Chinese Academy of Sciences	Yes	Yes	No
ictnp3	Chinese Academy of Sciences	Yes	Yes	No
ictnp4	Chinese Academy of Sciences	Yes	Yes	No
ictnp6	Chinese Academy of Sciences	Yes	Yes	No
ictnp7	Chinese Academy of Sciences	Yes	Yes	No
pltr02wt6	City University – London	No	No	No
pltr02wt7	City University – London	No	No	No
pltr02wt8	City University – London	No	No	No
pltr02wt9	City University – London	No	No	No
csiro02np01	CSIRO	No	No	No
csiro02np02	CSIRO	No	No	No
csiro02np03	CSIRO	Yes	No	No
csiro02np04	CSIRO	Yes	Yes	No
csiro02np016	CSIRO	Yes	Yes	Yes
hum02pd	Hummingbird	Yes	No	No
hum02ud	Hummingbird	No	No	No
hum02uhp	Hummingbird	Yes	No	No
hum02up	Hummingbird	Yes	No	No
hum02upd	Hummingbird	Yes	No	No
iit02b	IIT Information Retrieval Lab	No	No	No
iit02tf	IIT Information Retrieval Lab	Yes	No	No
iit02fa	IIT Information Retrieval Lab	Yes	Yes	No
KUHPF0201	Kasetsart University	No	Yes	No
litlink	Laboratory for Information Technology (KRDL)	Yes	Yes	Yes
littext	Laboratory for Information Technology (KRDL)	No	No	No
thunp1	Tsinghua University	Yes	Yes	No
thunp2	Tsinghua University	Yes	Yes	No
thunp3	Tsinghua University	Yes	Yes	No
thunp4	Tsinghua University	Yes	Yes	No
thunp5	Tsinghua Unversity	Yes	Yes	No
pirc2Wnp1	Queens College, CUNY	Yes	Yes	No
picr2Wnp2	Queens College, CUNY	Yes	Yes	Yes
UAmsT02WnA	University of Amsterdam-Monz	No	Yes	No
UAmsT02WnTl	University of Amsterdam-Monz	No	No	No
UAmsT02WnTlA	University of Amsterdam-Monz	No	Yes	No
UAmsT02WnTm	University of Amsterdam-Monz	No	No	No
UAmsT02WnTmA	University of Amsterdam-Monz	No	Yes	No

Web Track, Named Page Finding Task (continued)

uog06c	University of Glasgow	No	No	No
uog07cta	University of Glasgow	Yes	Yes	No
uog08ctap	University of Glasgow	Yes	Yes	No
uog09cta2	University of Glasgow	Yes	Yes	No
uog10ctad	University of Glasgow	Yes	Yes	Yes
uicnp01	University of Illinois at Chicago	No	No	Yes
uicnp02	University of Illinois at Chicago	No	No	No
uicnp03	University of Illinois at Chicago	No	No	No
MU106	The University of Melbourne	Yes	Yes	No
MU208	The University of Melbourne	Yes	Yes	No
MU307	The University of Melbourne	Yes	Yes	No
MU609	The University of Melbourne	Yes	Yes	No
MU80A	The University of Melbourne	No	No	No
UniNEnp1	University of Neuchatel	Yes	Yes	No
UniNEnp2	University of Neuchatel	Yes	Yes	No
UniNEnp3	University of Neuchatel	Yes	Yes	No
UniNEnp4	University of Neuchatel	Yes	Yes	No
uwmtBW0	University of Waterloo	No	No	No
uwmtBW1	University of Waterloo	No	No	Yes
uwmtBW2	University of Waterloo	No	Yes	Yes
uwmtBW3	University of Waterloo	No	Yes	Yes
uwmtBW4	University of Waterloo	No	Yes	Yes
yedi01no	Yonsei Univ. and ETRI	Yes	Yes	Yes
yedi01	Yonsei Univ. and ETRI	Yes	Yes	Yes
yenp01	Yonsei Univ. and ETRI	Yes	Yes	Yes

Web Track, Topic Distillation Task

Tag	Organization	Document Structure	Anchor Text	Link Structure
ajouai0206	Ajou Univ.	No	No	Yes
ajouai0207	Ajou Univ.	No	No	Yes
ajouai0208	Ajou Univ.	No	No	Yes
ajouai0209	Ajou Univ.	No	No	Yes
ajouai0210	Ajou Univ.	No	No	Yes
icttd1	Chinese Academy of Sciences	No	No	No
icttd2	Chinese Academy of Sciences	No	No	No
icttd3	Chinese Academy of Sciences	No	No	No
pltr02wt1	City Univ.-London	No	No	No
pltr02wt2	City Univ.-London	No	No	No
pltr02wt3	City Univ.-London	No	No	No
pltr02wt4	City Univ.-London	No	No	No
pltr02wt5	City Univ.-London	No	No	No
csiro02td1	CSIRO	No	No	Yes
csiro02td2	CSIRO	No	Yes	Yes
csiro02td3	CSIRO	No	No	Yes
csiro02td4	CSIRO	No	Yes	Yes
csiro02td5	CSIRO	No	No	Yes
fduwt11b0	Fudan Univ.	No	No	No
fduwt11o1	Fudan Univ.	Yes	No	Yes
fduwt11o2	Fudan Univ.	Yes	Yes	Yes
fduwt11t1	Fudan Univ.	Yes	No	Yes
fduwt11t2	Fudan Univ.	Yes	Yes	Yes
ibmHaifaAP	IBM Research Lab in Haifa	Yes	Yes	Yes
ibmHaifaBase	IBM Research Lab in Haifa	Yes	Yes	Yes
ibmHaifaPR	IBM Research Lab in Haifa	Yes	Yes	Yes
ibmHaifaT10	IBM Research Lab in Haifa	Yes	Yes	Yes
ibmHaifaT10D	IBM Research Lab in Haifa	Yes	Yes	Yes
Mercah	IRIT/SIG	No	No	No
Mercure	IRIT/SIG	No	No	Yes
MercureLynx	IRIT/SIG	No	No	Yes
thutd1	Tsinghua Univ.	Yes	Yes	Yes
thutd2	Tsinghua Univ.	Yes	Yes	No
thutd3	Tsinghua Univ.	Yes	Yes	No
thutd4	Tsinghua Univ.	Yes	Yes	No
thutd5	Tsinghua Univ.	Yes	Yes	No
pirc2Wd1	Queens College, CUNY	Yes	Yes	Yes
pirc2Wd2	Queens College, CUNY	Yes	Yes	Yes
UAmsT02WtA	Univ. of Amsterdam-Monz	No	Yes	No
UAmsT02WtAcs	Univ. of Amsterdam-Monz	No	Yes	No
UAmsT02WtAri	Univ. of Amsterdam-Monz	No	Yes	Yes
UAmsT02WtT	Univ. of Amsterdam-Monz	No	No	No
UAmsT02WtTri	Univ. of Amsterdam-Monz	No	No	Yes
uog01ctaialh	Univ. of Glasgow	Yes	Yes	Yes
uog02ctadh	Univ. of Glasgow	Yes	Yes	Yes
uog03ctadqh	Univ. of Glasgow	Yes	Yes	Yes
uog04cta2dqh	Univ. of Glasgow	Yes	Yes	Yes
uog05tad	Univ. of Glasgow	Yes	Yes	Yes

Web Track, Topic Distillation Task (continued)

uic0101	Univ. of Illinois at Chicago	No	No	Yes
uic0102	Univ. of Illinois at Chicago	No	No	Yes
uic0103	Univ. of Illinois at Chicago	No	No	Yes
uic0104	Univ. of Illinois at Chicago	No	No	Yes
CARROT2A	Univ. of Md., Balt. Co.-Cost	No	No	No
CARROT2B	Univ. of Md., Balt. Co.-Cost	No	No	No
CARROT2C	Univ. of Md., Balt. Co.-Cost	No	No	Yes
CARROT2D	Univ. of Md., Balt. Co.-Cost	No	No	Yes
CARROT2E	Univ. of Md., Balt. Co.-Cost	No	No	Yes
MU111	Univ. of Melbourne	Yes	Yes	No
MU212	Univ. of Melbourne	Yes	Yes	No
MU313	Univ. of Melbourne	Yes	Yes	No
MU525	Univ. of Melbourne	No	No	No
MU624	Univ. of Melbourne	No	No	No
UniNEdi1	Univ. of Neuchatel	Yes	Yes	Yes
UniNEdi2	Univ. of Neuchatel	Yes	Yes	Yes
UniNEdi3	Univ. of Neuchatel	Yes	Yes	Yes
UniNEdi4	Univ. of Neuchatel	Yes	Yes	No
UniNEdi5	Univ. of Neuchatel	Yes	Yes	No
TDtfidf	Univ. of Sunderland-Stokoe	No	No	No
TDwsdtfidf	Univ. of Sunderland-Stokoe	No	No	No

1 Common Evaluation Measures

- Recall
  A measure of the ability of a system to present all relevant items.

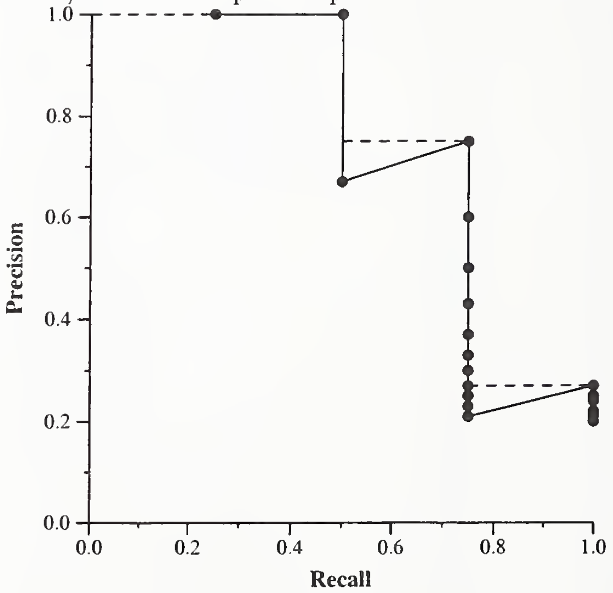
$$\text{recall} = \frac{\text{number of relevant items retrieved}}{\text{number of relevant items in collection}}$$

- Precision.
  A measure of the ability of a system to present only relevant items.

$$\text{precision} = \frac{\text{number of relevant items retrieved}}{\text{total number of items retrieved}}$$

Precision and recall are set-based measures. That is, they evaluate the quality of an unordered set of retrieved documents. To evaluate ranked lists, precision can be plotted against recall after each retrieved document as shown in the example below. To facilitate computing average performance over a set of topics— each with a different number of relevant documents— individual topic precision values are interpolated to a set of standard recall levels (0 to 1 in increments of .1). The particular rule used to interpolate precision at standard recall level i is to use the maximum precision obtained for the topic for any actual recall level greater than or equal to i. Note that while precision is not defined at a recall of 0.0, this interpolation rule does define an interpolated value for recall level 0.0. In the example, the actual precision values are plotted with circles (and connected by a solid line) and the interpolated precision is shown with the dashed line.

> Example: Assume a document collection has 20 documents, four of which are relevant to topic t. Further assume a retrieval system ranks the relevant documents first, second, fourth, and fifteenth. The exact recall points are 0.25, 0.5, 0.75, and 1.0. Using the interpolation rule, the interpolated precision for all standard recall levels up to .5 is 1, the interpolated precision for recall levels .6 and .7 is .75, and the interpolated precision for recall levels .8 or greater is .27.



2 trec_eval Evaluation Report

The results from the cross-language track, the topic distillation task in the web track, and the routing task in the filtering track are ranked lists of documents. These lists are evaluated using `trec_eval`, a program written by Chris Buckley when he was at Cornell University that can be obtained by anonymous ftp from Cornell in the directory pub/smart at ftp.cs.cornell.edu. An evaluation report for a run evaluated by `trec_eval` is comprised of a header (containing the task and organization name), 3 tables, and 2 graphs as described below. The feature and search tasks in the video track are also ranked list tasks that were evaluated with `trec_eval`, though no averages are reported for the feature task and the output has been relabeled.

2.1 Tables

I. "Summary Statistics" Table
Table 1 is a sample "Summary Statistics" Table

Table 1: Sample "Summary Statistics" Table.

Summary Statistics	
Run	Cor7A1clt–automatic, title
Number of Topics	50
Total number of documents over all topics	
Retrieved:	50000
Relevant:	4674
Rel_ret:	2621

A. Run
A description of the run. It contains the run tag provided by the participant, and various details about the runs such as whether queries were constructed manually or automatically.

B. Number of Topics
Number of topics searched in this run (generally 50 topics are run for each task).

C. Total number of documents over all topics (the number of topics given in B).

i. Retrieved
Number of documents submitted to NIST. This is usually 50,000 (50 topics × 1000 documents), but is less when fewer than 1000 documents are retrieved per topic.

ii. Relevant
Total possible relevant documents within a given task and category.

iii. Rel_ret
Total number of relevant documents returned by a run over all the topics.

II. "Recall Level Precision Averages" Table.
Table 2 is a sample "Recall Level Precision Averages" Table.

A. Precision at 11 standard recall levels
The precision averages at 11 standard recall levels are used to compare the performance of different systems and as the input for plotting the recall-precision graph (see below). Each recall-precision average is computed by summing the interpolated precisions at the specified recall cutoff value (denoted by $\sum P_\lambda$ where P_λ is the interpolated precision at recall level λ) and then dividing by the number of topics.

$$\frac{\sum_{i=1}^{NUM} P_\lambda}{NUM} \qquad \lambda = \{0.0, 0.1, 0.2, 0.3, \ldots, 1.0\}$$

Table 2: Sample "Recall Level Precision Averages" Table.

Recall Level Precision Averages	
Recall	Precision
0.00	0.6169
0.10	0.4517
0.20	0.3938
0.30	0.3243
0.40	0.2715
0.50	0.2224
0.60	0.1642
0.70	0.1342
0.80	0.0904
0.90	0.0472
1.00	0.0031
Average precision over all relevant docs	
non-interpolated	0.2329

- Interpolating recall-precision
  Standard recall levels facilitate averaging and plotting retrieval results.

B. Average precision over all relevant documents, non-interpolated
This is a single-valued measure that reflects the performance over all relevant documents. It rewards systems that retrieve relevant documents quickly (highly ranked).

The measure is not an average of the precision at standard recall levels. Rather, it is the average of the precision value obtained after each relevant document is retrieved. (When a relevant document is not retrieved at all, its precision is assumed to be 0.) As an example, consider a query that has four relevant documents which are retrieved at ranks 1, 2, 4, and 7. The actual precision obtained when each relevant document is retrieved is 1, 1, 0.75, and 0.57, respectively, the mean of which is 0.83. Thus, the average precision over all relevant documents for this query is 0.83.

III. "Document Level Averages" Table
Table 3 is a sample "Document Level Averages" Table.

Table 3: Sample "Document Level Averages" Table.

Document Level Averages	
	Precision
At 5 docs	0.4280
At 10 docs	0.3960
At 15 docs	0.3493
At 20 docs	0.3370
At 30 docs	0.3100
At 100 docs	0.2106
At 200 docs	0.1544
At 500 docs	0.0875
At 1000 docs	0.0524
R–Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.2564

A. Precision at 9 document cutoff values

The precision computed after a given number of documents have been retrieved reflects the actual measured system performance as a user might see it. Each document precision average is computed by summing the precisions at the specified document cutoff value and dividing by the number of topics (50).

B. R-Precision

R-Precision is the precision after R documents have been retrieved, where R is the number of relevant documents for the topic. It de-emphasizes the exact ranking of the retrieved relevant documents, which can be particularly useful in TREC where there are large numbers of relevant documents.

The average R-Precision for a run is computed by taking the mean of the R-Precisions of the individual topics in the run. For example, assume a run consists of two topics, one with 50 relevant documents and another with 10 relevant documents. If the retrieval system returns 17 relevant documents in the top 50 documents for the first topic, and 7 relevant documents in the top 10 for the second topic, then the run's R-Precision would be $\frac{\frac{17}{50}+\frac{7}{10}}{2}$ or 0.52.

2.2 Graphs

I. Recall-Precision Graph

Figure 1 is a sample Recall-Precision Graph.

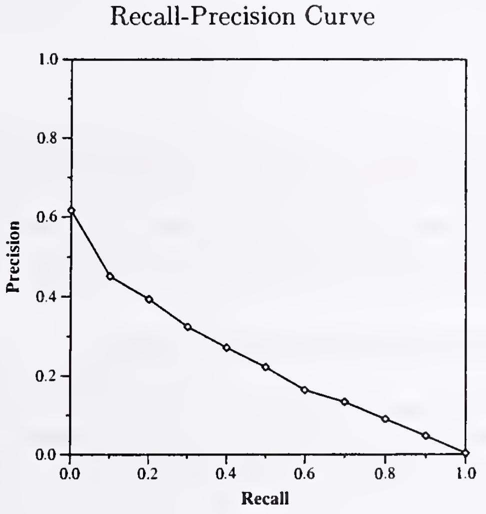


Figure 1: Sample Recall-Precision Graph.

The Recall-Precision Graph is created using the 11 cutoff values from the Recall Level Precision Averages. Typically these graphs slope downward from left to right, enforcing the notion that as more relevant documents are retrieved (recall increases), the more nonrelevant documents are retrieved (precision decreases).

This graph is the most commonly used method for comparing systems. The plots of different runs can be superimposed on the same graph to determine which run is superior. Curves closest to the upper right-hand corner of the graph (where recall and precision are maximized) indicate the best performance. Comparisons are best made in three different recall ranges: 0 to 0.2, 0.2 to 0.8, and 0.8 to 1. These ranges characterize high precision, middle recall, and high recall performance, respectively.

II. Average Precision Histogram.
Figure 2 is a sample Average Precision Histogram.

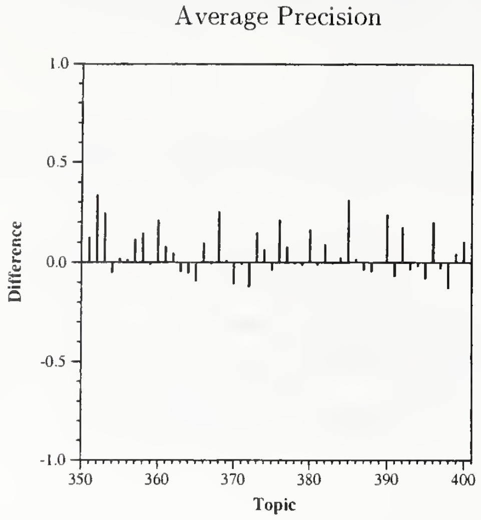


Figure 2: Sample Average Precision Histogram.

The Average Precision Histogram measures the average precision of a run on each topic against the median average precision of all corresponding runs on that topic. This graph is intended to give insight into the performance of individual systems and the types of topics that they handle well.

(Since the emphasis in the topic distillation task in the web track is high precision, the median histogram for that task uses Precision at a cut-off of 10, rather than average precision, as the underlying measure.)

3 Question Answering Evaluation Report

The tasks in the question answering track used different evaluation metrics and have different evaluation reports. For both tasks, responses were judged as being wrong, inexact, unsupported, or right. Right responses were considered correct, and all other responses as incorrect.

3.1 Main task

The primary evaluation measure used in the main task of the QA track is called the "confidence-weighted" score. This measure was inspired by the average uninterploated precision measure of ranked retrieval. Systems returned exactly one response for each of 500 questions, where the *questions* were ranked by the system's confidence in its response. That is, the question the system was most sure it answered correctly was ranked first and the question it was least sure it had answered correctly was ranked last. The confidence weighted score is then computed as

$$\frac{1}{500}\sum_{i=1}^{500}\frac{\text{number correct in first } i \text{ ranks}}{i}.$$

The evaluation report for the main task consists of a table giving details for the run and a graph that plots the confidence-weighted score versus question rank. An example of the table is shown in Table 4. The data given in the table include the raw counts of the number of questions judged in each category and the final confidence-weighted score. Also included are the precision and recall for the system's ability to

Table 4: Sample QA Main Task Table.

Summary Statistics	
Run ID	LCCmain2002
Num questions	500
Number wrong	63
Number unsupported	14
Number inexact	8
Number right	415
Confidence-weighted score	0.856
Number of times NIL correctly returned	38
Precision of recognizing no answer	37/64 = .0578
Recall of recognizing no answer	37/46 = 0.804

recognize when there was no correct answer in the collection. Precision of recognizing no answer is defined as the ratio of the number of times NIL was returned and correct to the number of times it was returned; recall as the ratio of the number of times NIL was returned and correct to the number of times it was correct (46).

The graph of confidence-weighted score versus question rank was created by plotting the confidence-weighted score computed up to rank i against i. It presents a graphical depiction of how well the system ranked the questions.

3.2 List task

The evaluation metric used for the list task is mean accuracy, where the accuracy of a single question is the number of distinct instances retrieved divided by the target number of instances (i.e., the number of instances the question specified should be retrieved). The evaluation report gives the run's mean accuracy computed over the 25 question in the test set. Also included is a histogram that shows the difference between the system's accuracy score and the median accuracy score for each question.

4 Filtering Evaluation Report

The result of a filtering run is an unordered set of documents, so it cannot be evaluated using `trec_eval`. (Routing runs do produce a ranked list of documents and are thus evaluated using `trec_eval`.) The evaluation measures used in the TREC 2002 filtering track were a normalized, scaled linear utility function and a variant of F-beta. If R^+ is the number of relevant documents a run retrieved, R^- the number of relevant documents that were not retrieved, and N^+ the number of non-relevant documents that were retrieved, the F-beta score used in the track is defined as

$$\text{T11F} = \begin{cases} 0 & \text{if } R^+ = N^+ = 0 \\ \dfrac{1.25R^+}{.25R^- + N^+ + 1.25R^+} & \text{otherwise} \end{cases}$$

and the utility function as

$$\text{T11U} = 2R^+ - N^+.$$

Average utility over a set of topics is computed using scaled utilities. The averaging proceeds as follows:

- normalize an individual topic's utility score by the maximum possible utility for that topic (2*total-relevant);
- scale that value according to a minimum acceptable level (-0.5 for TREC-2002);

- average the scaled, normalized utilities

The evaluation report for an adaptive filtering run consists of a table giving run characteristics and summary measures, a table and plot of average utility scores over different time periods, and a median graph. The batch filtering report contains just the characteristics table and median graph. The median graph shows the difference between the run's evaluation score and the median score for each topic. The evaluation score is either the F-beta score or the utility score, depending on what the run was optimized for.

In adaptive filtering, systems can modify profiles based on relevance information of retrieved documents. One strategy is to have a "liberal" retrieval policy early in the process to gain more information and then become more stringent as more is learned. The time graph for adaptive runs plots average utility for four different time periods where time periods are labeled by the document identifiers that exist in the time period.

5 Named Page Evaluation Report

The result of a named page run is a ranked list of documents, but the named page task is a known-item search and thus is not evaluated using `trec_eval`. Instead, the runs are evaluated using the rank at which the first correct named page was retrieved. The evaluation report consists of a table of evaluation scores and a median graph.

An example table of evaluation scores is given in Table 5. The table contains a description of the run that specifies whether document structure was exploited in the run, whether anchor text was exploited in the run, and whether link structure was exploited in the run. The evaluation scores reported include:

Table 5: Sample Named Page Task Table.

Summary Statistics	
Run ID	thunp3
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.719
Num found at rank 1	94 (62.7%)
Num found in top 10	133 (88.7%)
Num not found in top 100	12 (8.0%)

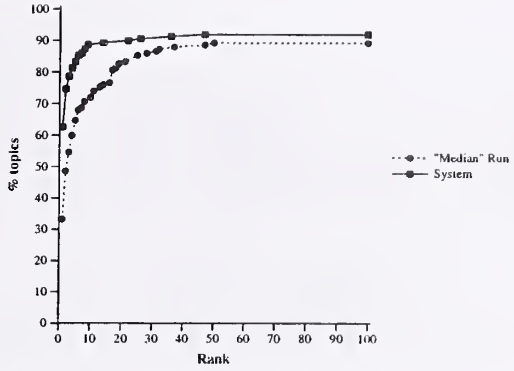
- The mean reciprocal rank for the run. The score for an individual topic is the reciprocal of the rank at which the first correct page was returned, or zero if no correct page was returned. The score for the run as a whole is the mean of the reciprocal rank over the test set of topics.

- The number and percentage of topics for which a correct page was retrieved in the first rank.

- The number and percentage of topics for which a correct page was retrieved in the top ten ranks (includes those topics for which the page was returned at rank one).

- The number and percentage of topics for which no correct page was returned in the top 100 ranks.

A sample median graph is shown in Figure 3. The graph plots the cumulative percentage of topics for which a correct page was retrieved by a given rank. Two lines are plotted, the results for the run, and the results for a hypothetical median run that retrieves the page at the median rank for each topic.



Cumulative % of topics that retrieve named page by given rank

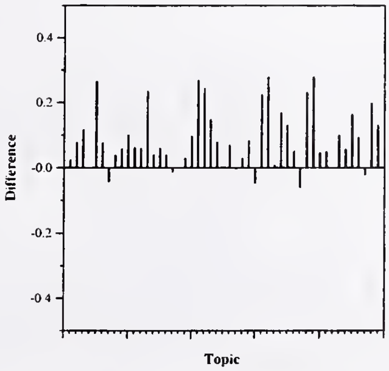
Figure 3: Sample Median Graph for the Named Page Task.

6 Novelty Track Evaluation Report

The task in the novelty track was divided into two parts, finding sentences that were relevant and then identifying a subset of those sentences that contained new information. The novelty track evaluation report contains the same set of measures reported twice, once for relevant sentences and once for new sentences.

The basic measures for the track are recall and precision. Let M be the number of matched sentences (i.e., the number of sentences selected by both the assessor and the system), A be the number of sentences selected by the assessor, and S be the number of sentences selected by the system. Then recall is M/A and precision is M/S. Because set precision and set recall do not average well, the primary measure used for the track is an F score with equal weighting of precision and recall: $\frac{2RP}{R+P}$.

The evaluation report consists of a table giving the average values for precision, recall, and F, plus a median graph. The median graph shows the per-topic difference between the run's F score and the median F score for that topic. A sample novelty track median graph is shown in Figure 4.



Difference from Median in F score per topic for new sentences

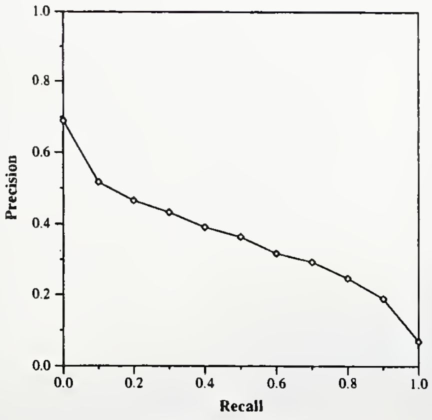
Figure 4: Sample median graph for the Novelty Track. This median graph is for the new sentence set.

Summary Statistics	
Run ID:	BBN11XLA
Run Description	English topics, automatic, title+desc
Number of Topics:	50

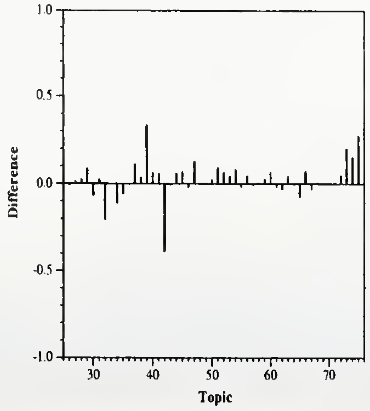
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	5177

Recall Level Averages	
Recall	Precision
0.00	0.6893
0.10	0.5171
0.20	0.4657
0.30	0.4329
0.40	0.3917
0.50	0.3646
0.60	0.3175
0.70	0.2934
0.80	0.2472
0.90	0.1899
1.00	0.0697
Mean average precision	
non-interpolated	0.3444

Document Level Averages	
	Precision
At 5 docs	0.4800
At 10 docs	0.4420
At 15 docs	0.4240
At 20 docs	0.4200
At 30 docs	0.3940
At 100 docs	0.3380
At 200 docs	0.2819
At 500 docs	0.1753
At 1000 docs	0.1035
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3584



Recall-Precision Curve



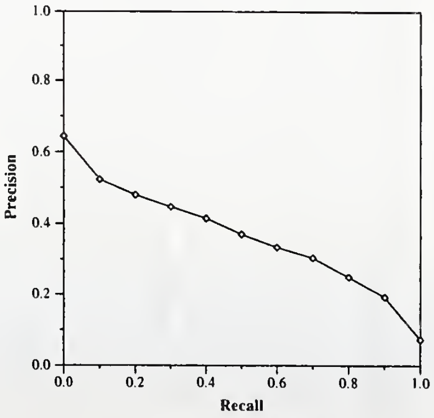
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BBN11XLB
Run Description	English topics, automatic, title+desc
Number of Topics:	50

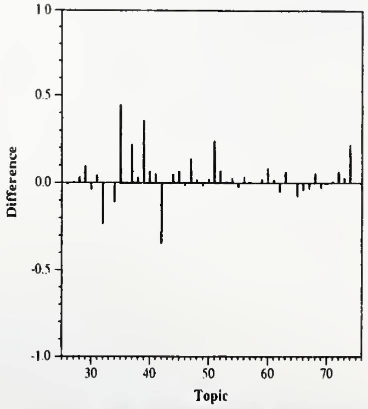
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	5215

Recall Level Averages	
Recall	Precision
0.00	0.6441
0.10	0.5224
0.20	0.4797
0.30	0.4469
0.40	0.4144
0.50	0.3702
0.60	0.3338
0.70	0.3038
0.80	0.2493
0.90	0.1932
1.00	0.0729
Mean average precision	
non-interpolated	0.3514

Document Level Averages	
	Precision
At 5 docs	0.4560
At 10 docs	0.4280
At 15 docs	0.4253
At 20 docs	0.4180
At 30 docs	0.3907
At 100 docs	0.3448
At 200 docs	0.2845
At 500 docs	0.1772
At 1000 docs	0.1043
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3652



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BBN11XLC
Run Description	English topics, automatic, title+desc
Number of Topics:	50

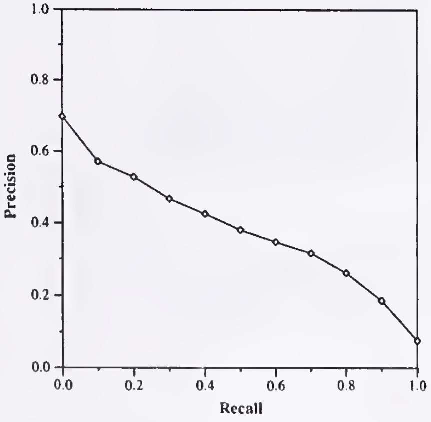
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	5053

Recall Level Averages	
Recall	Precision
0.00	0.6978
0.10	0.5713
0.20	0.5287
0.30	0.4678
0.40	0.4258
0.50	0.3812
0.60	0.3475
0.70	0.3164
0.80	0.2622
0.90	0.1869
1.00	0.0765

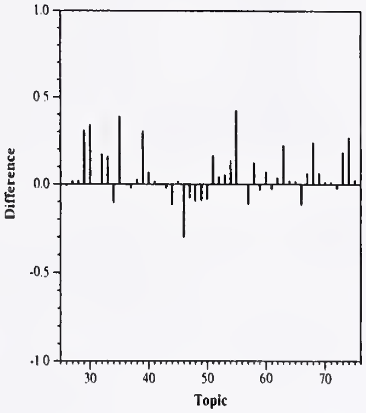
Mean average precision	
non-interpolated	0.3756

Document Level Averages	
	Precision
At 5 docs	0.5040
At 10 docs	0.4660
At 15 docs	0.4653
At 20 docs	0.4500
At 30 docs	0.4373
At 100 docs	0.3516
At 200 docs	0.2734
At 500 docs	0.1676
At 1000 docs	0.1011

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3736



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BBN11XLS
Run Description	English topics, automatic, title+desc
Number of Topics:	50

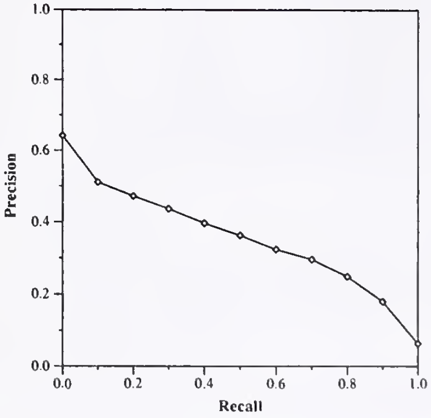
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	5057

Recall Level Averages	
Recall	Precision
0.00	0.6417
0.10	0.5106
0.20	0.4720
0.30	0.4366
0.40	0.3963
0.50	0.3625
0.60	0.3233
0.70	0.2955
0.80	0.2489
0.90	0.1801
1.00	0.0638

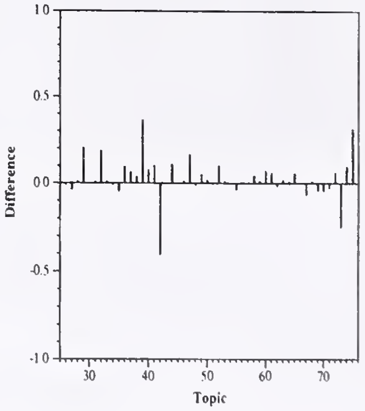
Mean average precision	
non-interpolated	0.3473

Document Level Averages	
	Precision
At 5 docs	0.4480
At 10 docs	0.4380
At 15 docs	0.4267
At 20 docs	0.4120
At 30 docs	0.3920
At 100 docs	0.3332
At 200 docs	0.2750
At 500 docs	0.1708
At 1000 docs	0.1011

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3573



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	humAR02td
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	50

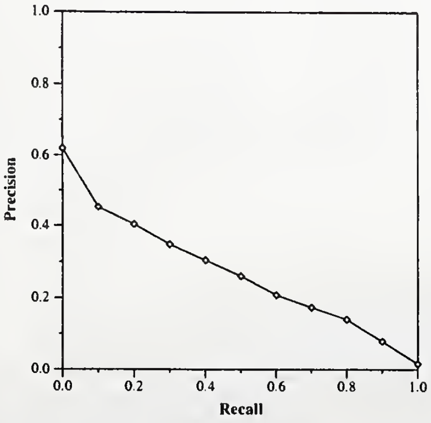
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4166

Recall Level Averages	
Recall	Precision
0.00	0.6194
0.10	0.4525
0.20	0.4041
0.30	0.3486
0.40	0.3046
0.50	0.2610
0.60	0.2085
0.70	0.1746
0.80	0.1411
0.90	0.0796
1.00	0.0163

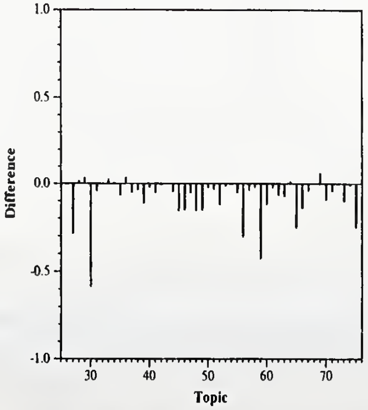
Mean average precision	
non-interpolated	0.2592

Document Level Averages	
	Precision
At 5 docs	0.3880
At 10 docs	0.3780
At 15 docs	0.3733
At 20 docs	0.3500
At 30 docs	0.3367
At 100 docs	0.2900
At 200 docs	0.2271
At 500 docs	0.1364
At 1000 docs	0.0833

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2949



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	humAR02tde
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	50

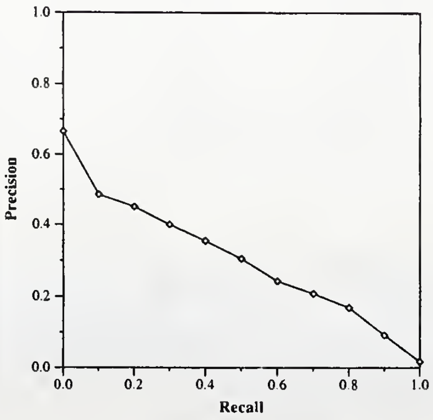
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4175

Recall Level Averages	
Recall	Precision
0.00	0.6657
0.10	0.4847
0.20	0.4501
0.30	0.3996
0.40	0.3535
0.50	0.3047
0.60	0.2423
0.70	0.2077
0.80	0.1689
0.90	0.0916
1.00	0.0168

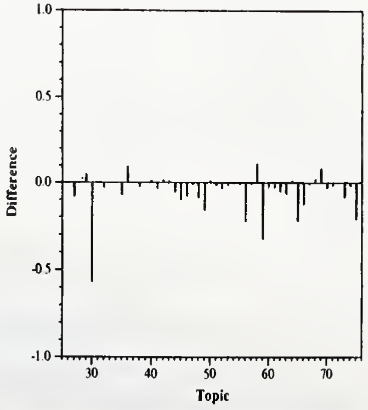
Mean average precision	
non-interpolated	0.2919

Document Level Averages	
	Precision
At 5 docs	0.3960
At 10 docs	0.4220
At 15 docs	0.4067
At 20 docs	0.3980
At 30 docs	0.3853
At 100 docs	0.3166
At 200 docs	0.2528
At 500 docs	0.1476
At 1000 docs	0.0835

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3272



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	humAR02tdm
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	50

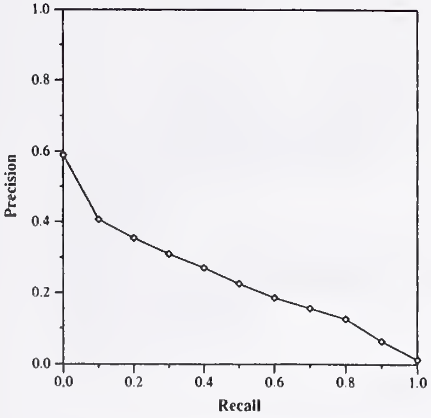
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4041

Recall Level Averages	
Recall	Precision
0.00	0.5894
0.10	0.4063
0.20	0.3542
0.30	0.3097
0.40	0.2706
0.50	0.2257
0.60	0.1862
0.70	0.1565
0.80	0.1262
0.90	0.0639
1.00	0.0128

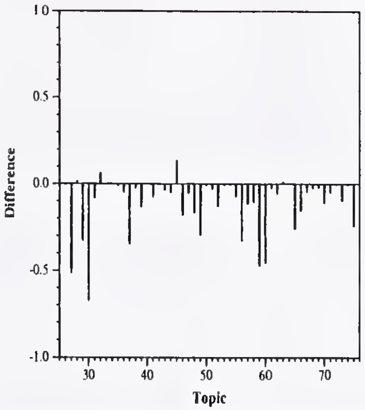
Mean average precision	
non-interpolated	0.2270

Document Level Averages	
	Precision
At 5 docs	0.3200
At 10 docs	0.3580
At 15 docs	0.3293
At 20 docs	0.3250
At 30 docs	0.3213
At 100 docs	0.2706
At 200 docs	0.2146
At 500 docs	0.1346
At 1000 docs	0.0808

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2736



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	humAR02tdne
Run Description	Arabic topics, automatic, title+desc+narr
Number of Topics:	50

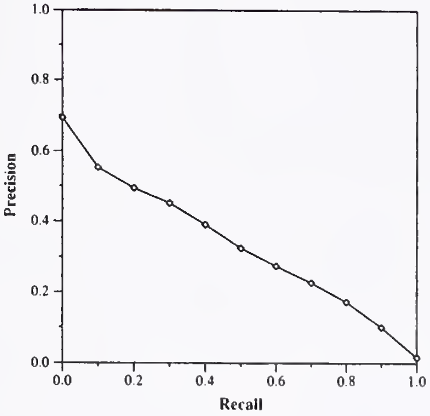
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4267

Recall Level Averages	
Recall	Precision
0.00	0.6936
0.10	0.5522
0.20	0.4935
0.30	0.4511
0.40	0.3902
0.50	0.3242
0.60	0.2741
0.70	0.2266
0.80	0.1727
0.90	0.1006
1.00	0.0160

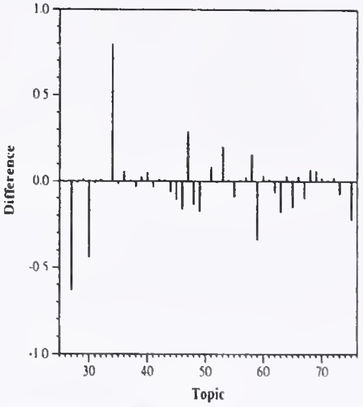
Mean average precision	
non-interpolated	0.3190

Document Level Averages	
	Precision
At 5 docs	0.4760
At 10 docs	0.4700
At 15 docs	0.4627
At 20 docs	0.4470
At 30 docs	0.4200
At 100 docs	0.3348
At 200 docs	0.2591
At 500 docs	0.1525
At 1000 docs	0.0853

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3532



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	humAR02te
Run Description	Arabic topics, automatic, title
Number of Topics:	50

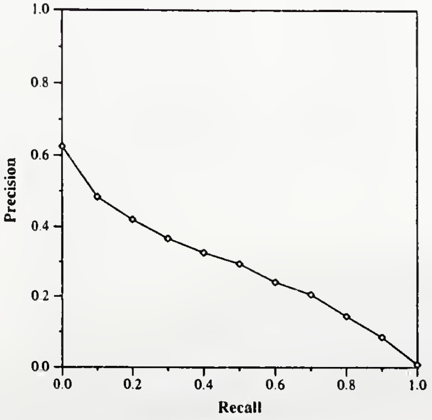
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4084

Recall Level Averages	
Recall	Precision
0.00	0.6245
0.10	0.4846
0.20	0.4205
0.30	0.3666
0.40	0.3263
0.50	0.2946
0.60	0.2415
0.70	0.2062
0.80	0.1442
0.90	0.0851
1.00	0.0079

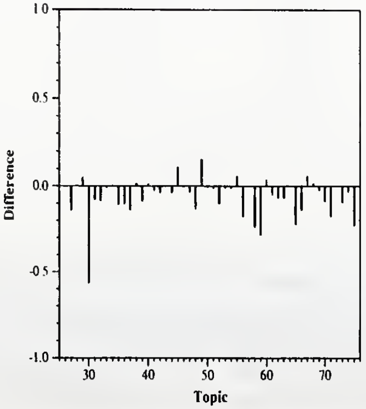
Mean average precision	
non-interpolated	0.2782

Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.3840
At 15 docs	0.3733
At 20 docs	0.3570
At 30 docs	0.3447
At 100 docs	0.3022
At 200 docs	0.2473
At 500 docs	0.1488
At 1000 docs	0.0817

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3105



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ibmy02d
Run Description	Arabic topics, automatic, title
Number of Topics:	50

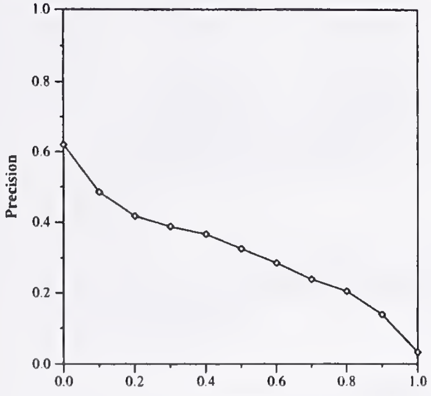
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4534

Recall Level Averages	
Recall	Precision
0.00	0.6199
0.10	0.4848
0.20	0.4175
0.30	0.3882
0.40	0.3668
0.50	0.3258
0.60	0.2853
0.70	0.2392
0.80	0.2050
0.90	0.1387
1.00	0.0325

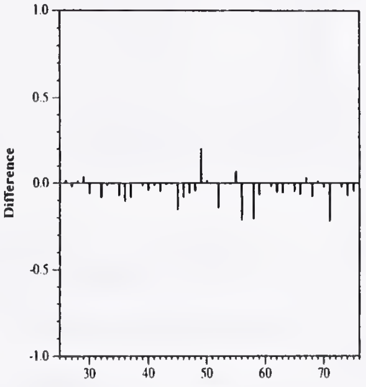
Mean average precision	
non-interpolated	0.3030

Document Level Averages	
	Precision
At 5 docs	0.3800
At 10 docs	0.3960
At 15 docs	0.4040
At 20 docs	0.3820
At 30 docs	0.3580
At 100 docs	0.2914
At 200 docs	0.2484
At 500 docs	0.1545
At 1000 docs	0.0907

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3335



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ibmy02a
Run Description	English topics, automatic, title+desc
Number of Topics:	50

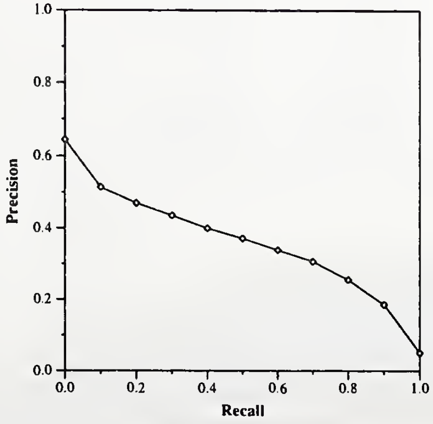
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	5152

Recall Level Averages	
Recall	Precision
0.00	0.6444
0.10	0.5141
0.20	0.4698
0.30	0.4355
0.40	0.3999
0.50	0.3715
0.60	0.3388
0.70	0.3067
0.80	0.2553
0.90	0.1867
1.00	0.0507

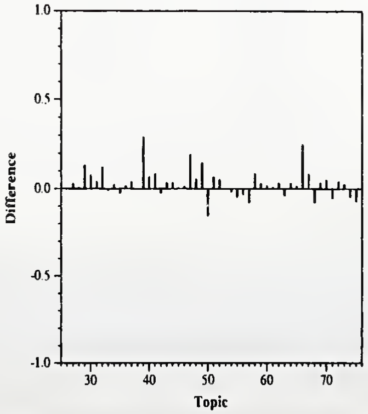
Mean average precision	
non-interpolated	0.3509

Document Level Averages	
	Precision
At 5 docs	0.4480
At 10 docs	0.4260
At 15 docs	0.4227
At 20 docs	0.4170
At 30 docs	0.4093
At 100 docs	0.3474
At 200 docs	0.2885
At 500 docs	0.1770
At 1000 docs	0.1030

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3578



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ibmy02b
Run Description	English topics, automatic, title+desc
Number of Topics:	50

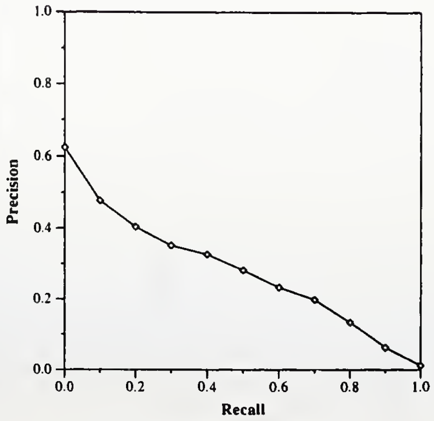
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4415

Recall Level Averages	
Recall	Precision
0.00	0.6240
0.10	0.4776
0.20	0.4042
0.30	0.3522
0.40	0.3267
0.50	0.2821
0.60	0.2334
0.70	0.1990
0.80	0.1342
0.90	0.0634
1.00	0.0117

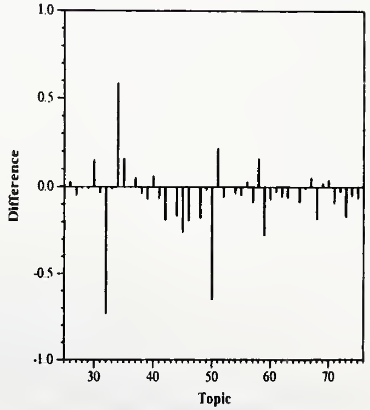
Mean average precision	
non-interpolated	0.2705

Document Level Averages	
	Precision
At 5 docs	0.4720
At 10 docs	0.4280
At 15 docs	0.4013
At 20 docs	0.3760
At 30 docs	0.3553
At 100 docs	0.2952
At 200 docs	0.2345
At 500 docs	0.1458
At 1000 docs	0.0883

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2979



Recall-Precision Curve



Difference from Median in Average Precision per Topic

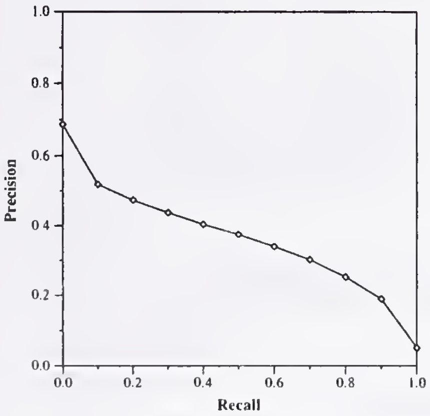
Cross-language track results — IBM T.J. Watson Research Center-Ittycheriah

Summary Statistics	
Run ID:	ibmy02c
Run Description	English topics, automatic, title+desc
Number of Topics:	50

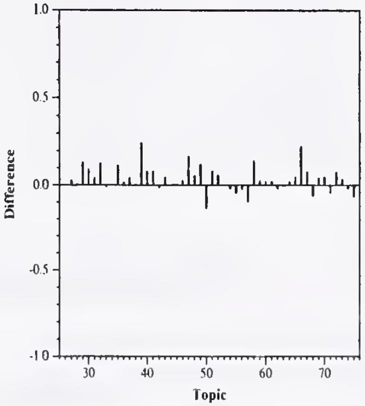
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	5162

Recall Level Averages	
Recall	Precision
0.00	0.6873
0.10	0.5186
0.20	0.4731
0.30	0.4373
0.40	0.4036
0.50	0.3744
0.60	0.3392
0.70	0.3014
0.80	0.2520
0.90	0.1897
1.00	0.0515
Mean average precision	
non-interpolated	0.3563

Document Level Averages	
	Precision
At 5 docs	0.4720
At 10 docs	0.4260
At 15 docs	0.4427
At 20 docs	0.4290
At 30 docs	0.4160
At 100 docs	0.3508
At 200 docs	0.2892
At 500 docs	0.1774
At 1000 docs	0.1032
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3610



Recall-Precision Curve



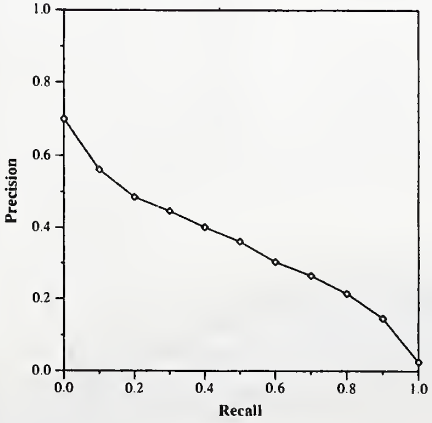
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	iit02mel
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	50

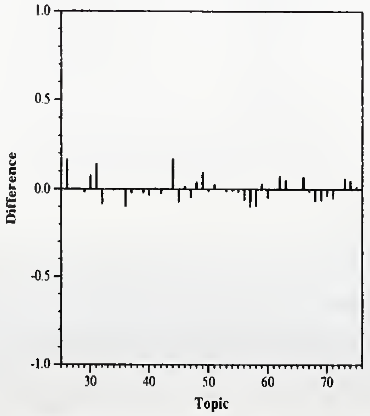
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4771

Recall Level Averages	
Recall	Precision
0.00	0.6993
0.10	0.5611
0.20	0.4854
0.30	0.4470
0.40	0.4014
0.50	0.3610
0.60	0.3032
0.70	0.2649
0.80	0.2149
0.90	0.1469
1.00	0.0255
Mean average precision	
non-interpolated	0.3419

Document Level Averages	
	Precision
At 5 docs	0.4840
At 10 docs	0.4260
At 15 docs	0.4107
At 20 docs	0.4000
At 30 docs	0.3753
At 100 docs	0.3262
At 200 docs	0.2725
At 500 docs	0.1654
At 1000 docs	0.0954
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3637



Recall-Precision Curve



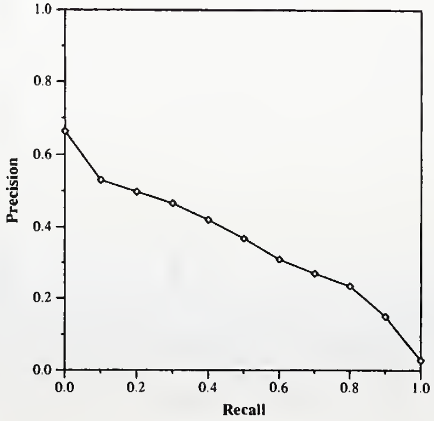
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	iit02mnl
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	50

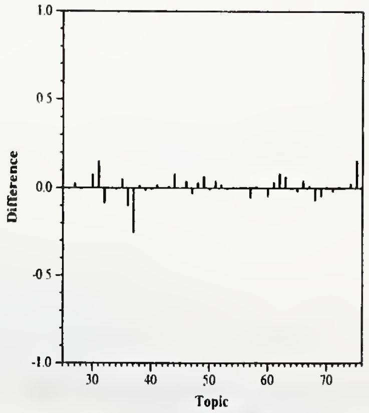
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4844

Recall Level Averages	
Recall	Precision
0.00	0.6638
0.10	0.5299
0.20	0.4978
0.30	0.4662
0.40	0.4200
0.50	0.3681
0.60	0.3088
0.70	0.2694
0.80	0.2342
0.90	0.1500
1.00	0.0285
Mean average precision	
non-interpolated	0.3454

Document Level Averages	
	Precision
At 5 docs	0.4560
At 10 docs	0.4440
At 15 docs	0.4160
At 20 docs	0.4020
At 30 docs	0.3787
At 100 docs	0.3334
At 200 docs	0.2747
At 500 docs	0.1679
At 1000 docs	0.0969
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3694



Recall-Precision Curve



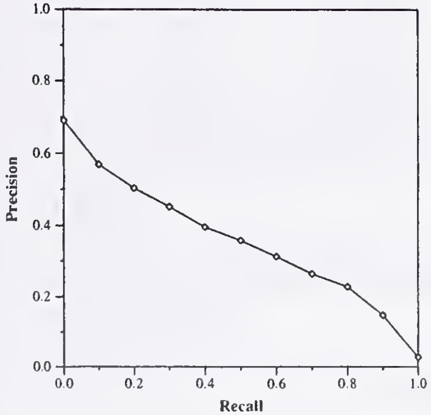
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	iit02mpl
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	50

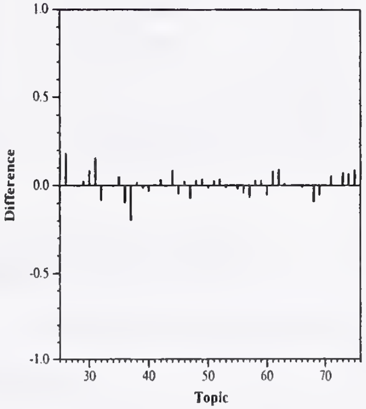
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4811

Recall Level Averages	
Recall	Precision
0.00	0.6913
0.10	0.5692
0.20	0.5031
0.30	0.4517
0.40	0.3957
0.50	0.3575
0.60	0.3125
0.70	0.2646
0.80	0.2284
0.90	0.1481
1.00	0.0273
Mean average precision	
non-interpolated	0.3473

Document Level Averages	
	Precision
At 5 docs	0.4760
At 10 docs	0.4620
At 15 docs	0.4347
At 20 docs	0.4110
At 30 docs	0.3920
At 100 docs	0.3332
At 200 docs	0.2711
At 500 docs	0.1673
At 1000 docs	0.0962
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3635



Recall-Precision Curve



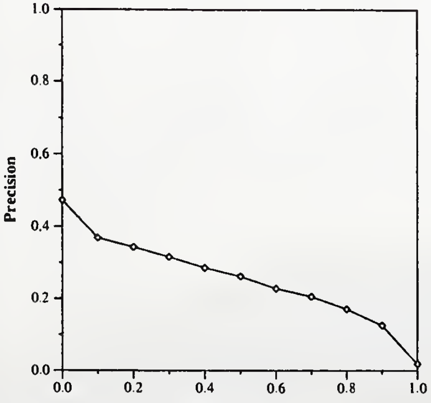
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	iit02xma
Run Description	English topics, automatic, title+desc
Number of Topics:	50

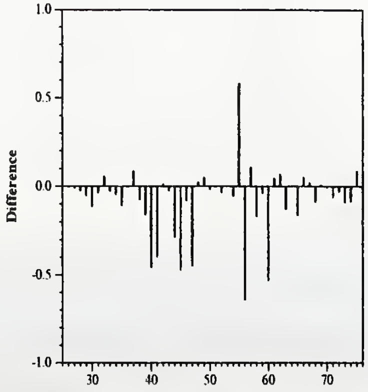
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	3597

Recall Level Averages	
Recall	Precision
0.00	0.4718
0.10	0.3690
0.20	0.3436
0.30	0.3164
0.40	0.2863
0.50	0.2624
0.60	0.2285
0.70	0.2060
0.80	0.1714
0.90	0.1262
1.00	0.0182
Mean average precision	
non-interpolated	0.2453

Document Level Averages	
	Precision
At 5 docs	0.3160
At 10 docs	0.3100
At 15 docs	0.3027
At 20 docs	0.2970
At 30 docs	0.2827
At 100 docs	0.2378
At 200 docs	0.2015
At 500 docs	0.1267
At 1000 docs	0.0719
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2583



Recall-Precision Curve



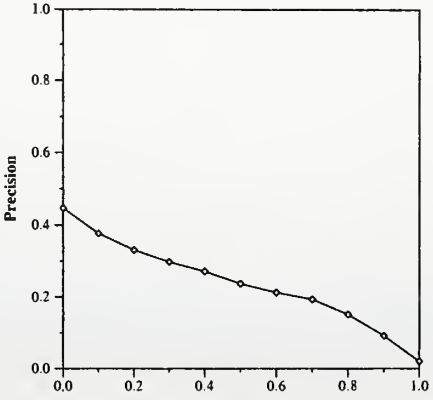
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	iit02xst
Run Description	English topics, automatic, title+desc
Number of Topics:	50

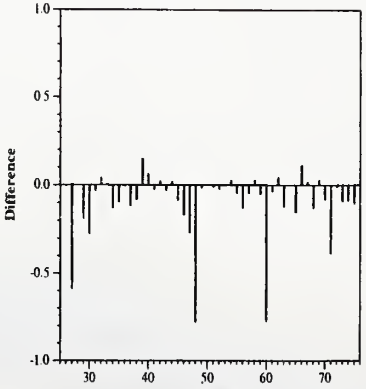
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4026

Recall Level Averages	
Recall	Precision
0.00	0.4463
0.10	0.3771
0.20	0.3314
0.30	0.2982
0.40	0.2717
0.50	0.2377
0.60	0.2127
0.70	0.1936
0.80	0.1513
0.90	0.0919
1.00	0.0205
Mean average precision	
non-interpolated	0.2285

Document Level Averages	
	Precision
At 5 docs	0.3040
At 10 docs	0.3240
At 15 docs	0.3027
At 20 docs	0.2890
At 30 docs	0.2827
At 100 docs	0.2560
At 200 docs	0.2112
At 500 docs	0.1330
At 1000 docs	0.0805
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2516



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	apl11ca1
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	50

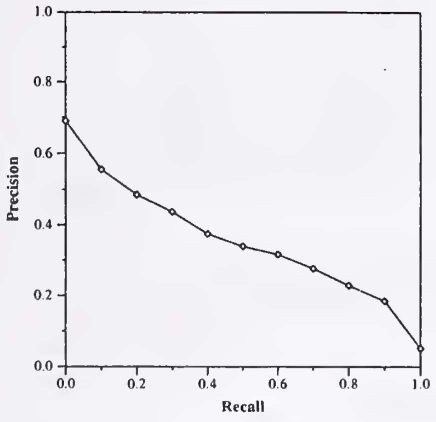
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4977

Recall Level Averages	
Recall	Precision
0.00	0.6903
0.10	0.5552
0.20	0.4861
0.30	0.4378
0.40	0.3750
0.50	0.3396
0.60	0.3163
0.70	0.2761
0.80	0.2284
0.90	0.1854
1.00	0.0520

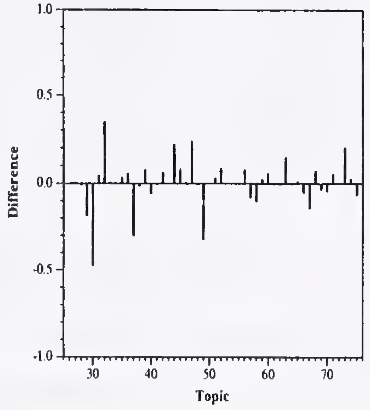
Mean average precision	
non-interpolated	0.3410

Document Level Averages	
	Precision
At 5 docs	0.4600
At 10 docs	0.4400
At 15 docs	0.4187
At 20 docs	0.4110
At 30 docs	0.3933
At 100 docs	0.3364
At 200 docs	0.2733
At 500 docs	0.1704
At 1000 docs	0.0995

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3530



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	apl11ce1
Run Description	English topics, automatic, title+desc
Number of Topics:	50

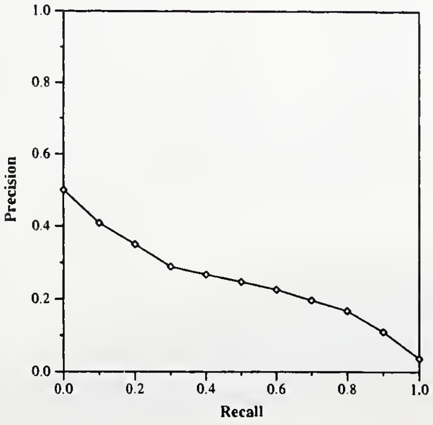
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4396

Recall Level Averages	
Recall	Precision
0.00	0.5003
0.10	0.4084
0.20	0.3500
0.30	0.2888
0.40	0.2671
0.50	0.2471
0.60	0.2257
0.70	0.1969
0.80	0.1680
0.90	0.1106
1.00	0.0368

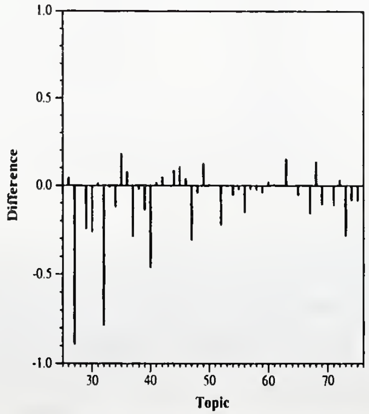
Mean average precision	
non-interpolated	0.2427

Document Level Averages	
	Precision
At 5 docs	0.3120
At 10 docs	0.3020
At 15 docs	0.3013
At 20 docs	0.3060
At 30 docs	0.2987
At 100 docs	0.2534
At 200 docs	0.2153
At 500 docs	0.1441
At 1000 docs	0.0879

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2631



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	apl11ce2
Run Description	English topics, automatic, title+desc
Number of Topics:	50

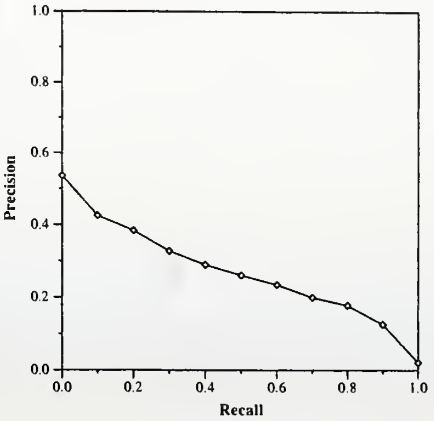
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4488

Recall Level Averages	
Recall	Precision
0.00	0.5358
0.10	0.4253
0.20	0.3836
0.30	0.3272
0.40	0.2891
0.50	0.2603
0.60	0.2341
0.70	0.1994
0.80	0.1781
0.90	0.1261
1.00	0.0211

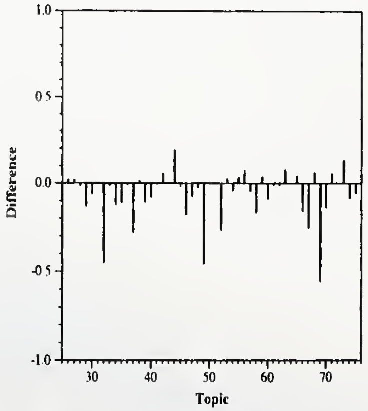
Mean average precision	
non-interpolated	0.2571

Document Level Averages	
	Precision
At 5 docs	0.3680
At 10 docs	0.3360
At 15 docs	0.3227
At 20 docs	0.3170
At 30 docs	0.3093
At 100 docs	0.2766
At 200 docs	0.2356
At 500 docs	0.1475
At 1000 docs	0.0898

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2714



Recall-Precision Curve



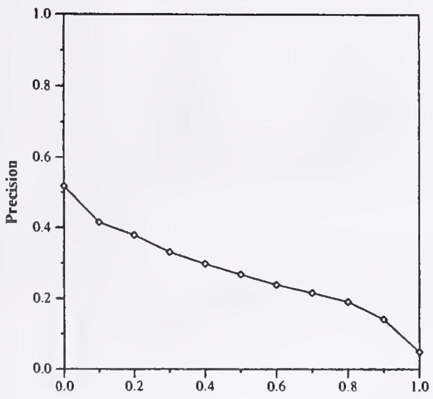
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	apl11ce3
Run Description	English topics, automatic, title+desc
Number of Topics:	50

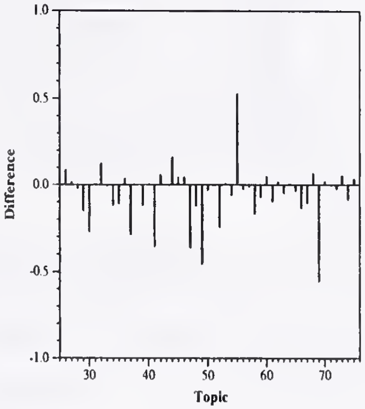
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4444

Recall Level Averages	
Recall	Precision
0.00	0.5175
0.10	0.4151
0.20	0.3789
0.30	0.3310
0.40	0.2987
0.50	0.2688
0.60	0.2398
0.70	0.2167
0.80	0.1910
0.90	0.1418
1.00	0.0487
Mean average precision	
non-interpolated	0.2658

Document Level Averages	
	Precision
At 5 docs	0.3560
At 10 docs	0.3400
At 15 docs	0.3333
At 20 docs	0.3250
At 30 docs	0.3127
At 100 docs	0.2616
At 200 docs	0.2158
At 500 docs	0.1425
At 1000 docs	0.0889
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2850



Recall-Precision Curve



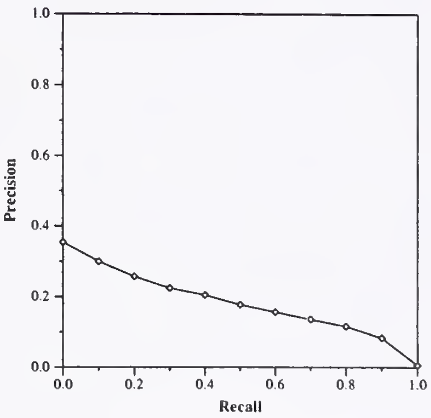
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	apl11ce4
Run Description	English topics, automatic, title+desc
Number of Topics:	50

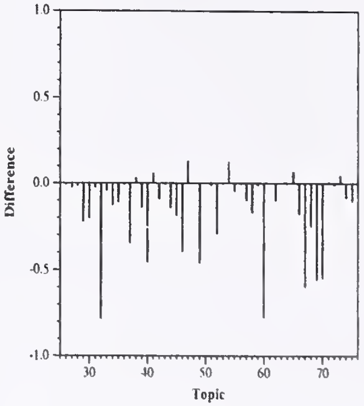
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	3645

Recall Level Averages	
Recall	Precision
0.00	0.3533
0.10	0.2996
0.20	0.2577
0.30	0.2251
0.40	0.2052
0.50	0.1776
0.60	0.1569
0.70	0.1363
0.80	0.1168
0.90	0.0836
1.00	0.0055
Mean average precision	
non-interpolated	0.1777

Document Level Averages	
	Precision
At 5 docs	0.2800
At 10 docs	0.2600
At 15 docs	0.2467
At 20 docs	0.2400
At 30 docs	0.2320
At 100 docs	0.2070
At 200 docs	0.1760
At 500 docs	0.1186
At 1000 docs	0.0729
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1904



Recall-Precision Curve



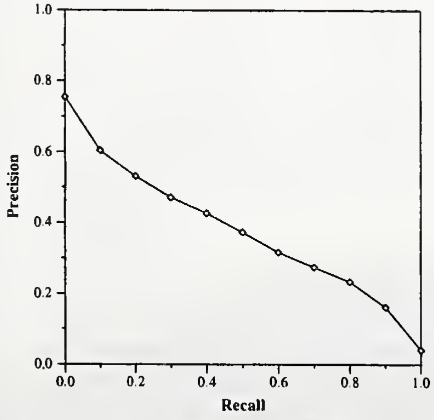
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BKYMON
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	50

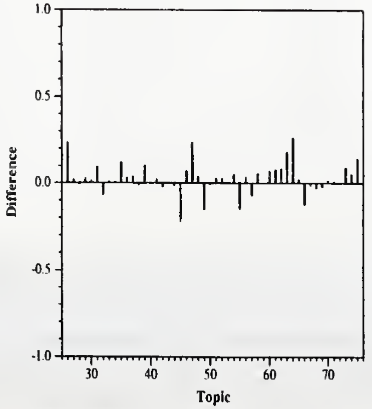
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4952

Recall Level Averages	
Recall	Precision
0.00	0.7552
0.10	0.6038
0.20	0.5307
0.30	0.4702
0.40	0.4261
0.50	0.3723
0.60	0.3159
0.70	0.2747
0.80	0.2337
0.90	0.1617
1.00	0.0406
Mean average precision	
non-interpolated	0.3666

Document Level Averages	
	Precision
At 5 docs	0.5480
At 10 docs	0.4880
At 15 docs	0.4480
At 20 docs	0.4290
At 30 docs	0.4033
At 100 docs	0.3452
At 200 docs	0.2744
At 500 docs	0.1681
At 1000 docs	0.0990
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3665



Recall-Precision Curve



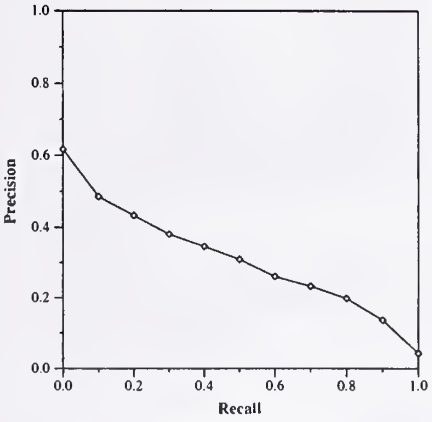
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BKYCL1
Run Description	English topics, automatic, title+desc
Number of Topics:	50

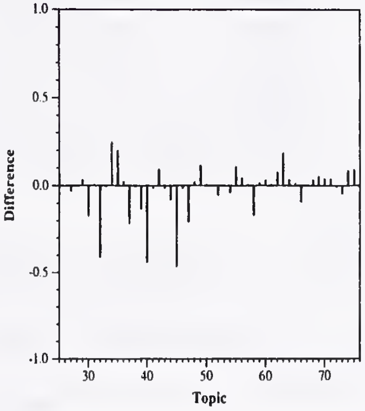
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4614

Recall Level Averages	
Recall	Precision
0.00	0.6164
0.10	0.4854
0.20	0.4331
0.30	0.3806
0.40	0.3457
0.50	0.3092
0.60	0.2605
0.70	0.2327
0.80	0.1976
0.90	0.1362
1.00	0.0421
Mean average precision	
non-interpolated	0.3000

Document Level Averages	
	Precision
At 5 docs	0.3960
At 10 docs	0.3960
At 15 docs	0.3720
At 20 docs	0.3550
At 30 docs	0.3393
At 100 docs	0.2914
At 200 docs	0.2457
At 500 docs	0.1582
At 1000 docs	0.0923
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3256



Recall-Precision Curve



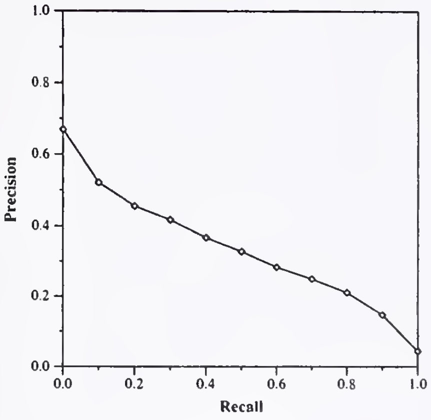
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BKYCL2
Run Description	English topics, automatic, title+desc
Number of Topics:	50

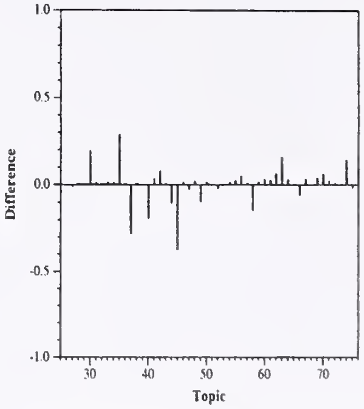
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4874

Recall Level Averages	
Recall	Precision
0.00	0.6696
0.10	0.5210
0.20	0.4553
0.30	0.4176
0.40	0.3668
0.50	0.3274
0.60	0.2832
0.70	0.2496
0.80	0.2107
0.90	0.1469
1.00	0.0434
Mean average precision	
non-interpolated	0.3224

Document Level Averages	
	Precision
At 5 docs	0.4480
At 10 docs	0.4280
At 15 docs	0.4093
At 20 docs	0.3890
At 30 docs	0.3653
At 100 docs	0.3170
At 200 docs	0.2628
At 500 docs	0.1643
At 1000 docs	0.0975
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3329



Recall-Precision Curve



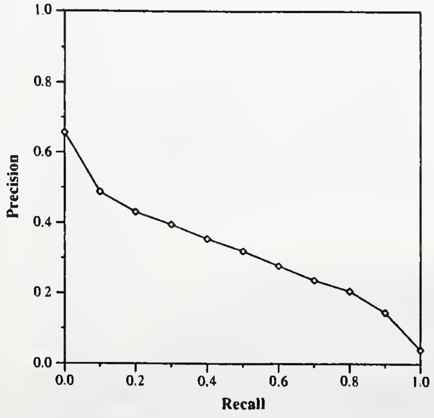
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	BKYCL3
Run Description	English topics, automatic, title+desc
Number of Topics:	50

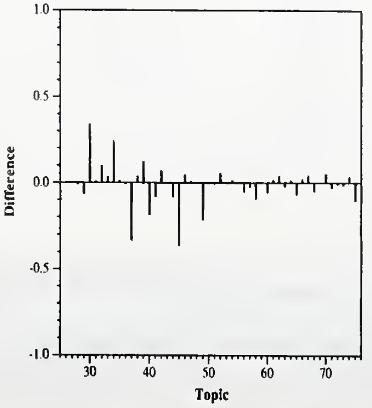
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4856

Recall Level Averages	
Recall	Precision
0.00	0.6567
0.10	0.4882
0.20	0.4309
0.30	0.3950
0.40	0.3552
0.50	0.3202
0.60	0.2786
0.70	0.2380
0.80	0.2064
0.90	0.1460
1.00	0.0399
Mean average precision	
non-interpolated	0.3089

Document Level Averages	
	Precision
At 5 docs	0.4480
At 10 docs	0.4080
At 15 docs	0.3920
At 20 docs	0.3800
At 30 docs	0.3653
At 100 docs	0.3060
At 200 docs	0.2516
At 500 docs	0.1600
At 1000 docs	0.0971
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3206



Recall-Precision Curve



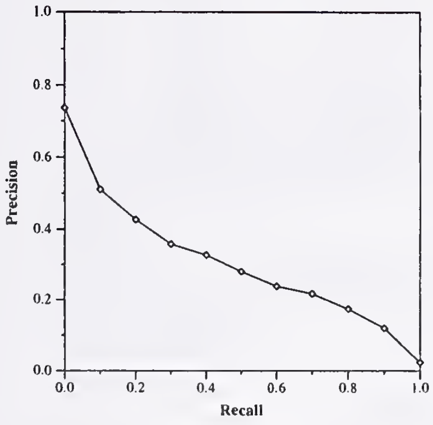
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	AutoMono
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	50

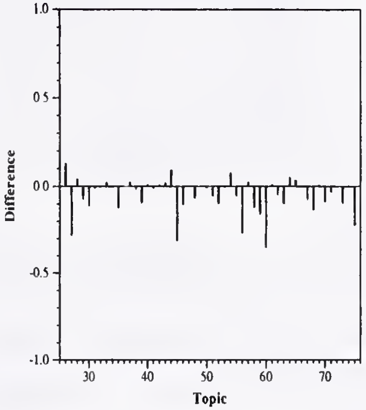
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4490

Recall Level Averages	
Recall	Precision
0.00	0.7359
0.10	0.5109
0.20	0.4271
0.30	0.3582
0.40	0.3273
0.50	0.2798
0.60	0.2376
0.70	0.2168
0.80	0.1739
0.90	0.1202
1.00	0.0236
Mean average precision	
non-interpolated	0.2891

Document Level Averages	
	Precision
At 5 docs	0.4440
At 10 docs	0.4080
At 15 docs	0.3853
At 20 docs	0.3760
At 30 docs	0.3500
At 100 docs	0.2854
At 200 docs	0.2357
At 500 docs	0.1505
At 1000 docs	0.0898
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3188



Recall-Precision Curve



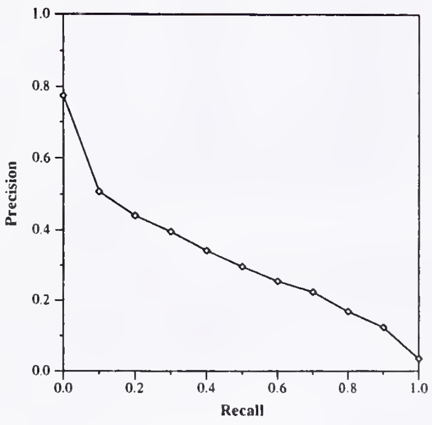
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	ManMono
Run Description	Arabic topics, manual
Number of Topics:	50

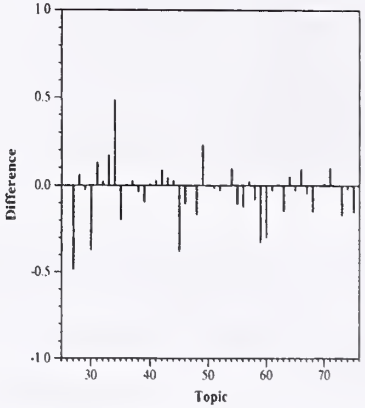
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4433

Recall Level Averages	
Recall	Precision
0.00	0.7742
0.10	0.5073
0.20	0.4405
0.30	0.3957
0.40	0.3422
0.50	0.2965
0.60	0.2548
0.70	0.2237
0.80	0.1686
0.90	0.1236
1.00	0.0350
Mean average precision	
non-interpolated	0.3019

Document Level Averages	
	Precision
At 5 docs	0.4840
At 10 docs	0.4340
At 15 docs	0.4093
At 20 docs	0.4010
At 30 docs	0.3680
At 100 docs	0.2936
At 200 docs	0.2349
At 500 docs	0.1474
At 1000 docs	0.0887
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3268



Recall-Precision Curve



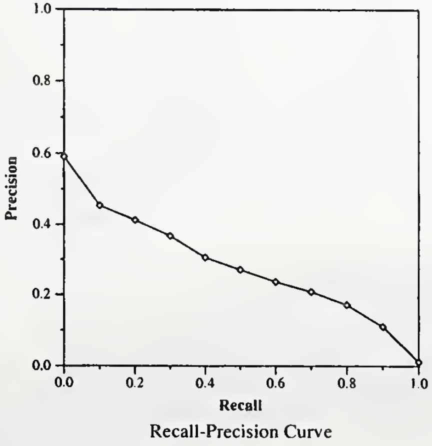
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	AutoClirDoc
Run Description	English topics, automatic, title+desc
Number of Topics:	50

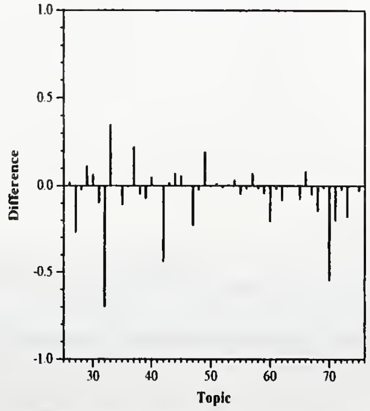
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4420

Recall Level Averages	
Recall	Precision
0.00	0.5914
0.10	0.4545
0.20	0.4131
0.30	0.3675
0.40	0.3063
0.50	0.2713
0.60	0.2366
0.70	0.2088
0.80	0.1724
0.90	0.1111
1.00	0.0114
Mean average precision	
non-interpolated	0.2740

Document Level Averages	
	Precision
At 5 docs	0.4000
At 10 docs	0.3720
At 15 docs	0.3640
At 20 docs	0.3440
At 30 docs	0.3227
At 100 docs	0.2826
At 200 docs	0.2367
At 500 docs	0.1482
At 1000 docs	0.0884
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3013



Recall-Precision Curve



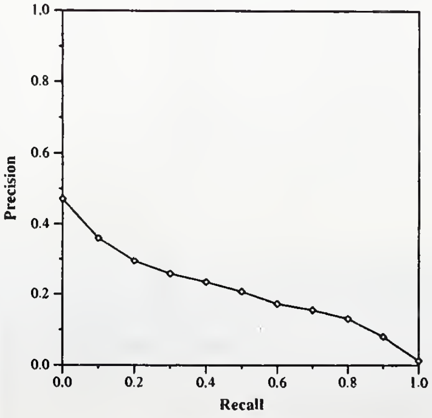
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	AutoClirExp
Run Description	English topics, automatic, title+desc
Number of Topics:	50

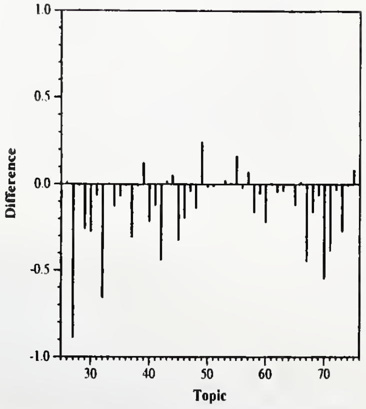
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	3722

Recall Level Averages	
Recall	Precision
0.00	0.4708
0.10	0.3591
0.20	0.2950
0.30	0.2586
0.40	0.2343
0.50	0.2072
0.60	0.1729
0.70	0.1556
0.80	0.1312
0.90	0.0803
1.00	0.0124
Mean average precision	
non-interpolated	0.2021

Document Level Averages	
	Precision
At 5 docs	0.2880
At 10 docs	0.2900
At 15 docs	0.2827
At 20 docs	0.2740
At 30 docs	0.2667
At 100 docs	0.2362
At 200 docs	0.1938
At 500 docs	0.1255
At 1000 docs	0.0744
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2429



Recall-Precision Curve



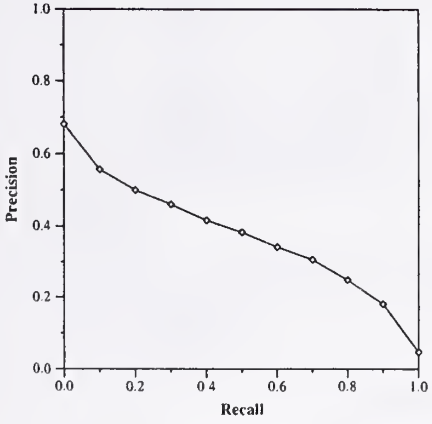
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UMassM
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	50

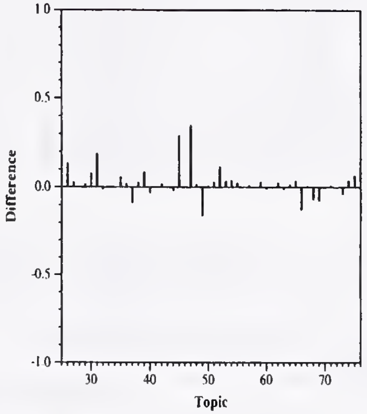
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	5093

Recall Level Averages	
Recall	Precision
0.00	0.6812
0.10	0.5567
0.20	0.4999
0.30	0.4607
0.40	0.4163
0.50	0.3831
0.60	0.3414
0.70	0.3058
0.80	0.2488
0.90	0.1810
1.00	0.0475
Mean average precision	
non-interpolated	0.3619

Document Level Averages	
	Precision
At 5 docs	0.4480
At 10 docs	0.4320
At 15 docs	0.4187
At 20 docs	0.4150
At 30 docs	0.4040
At 100 docs	0.3438
At 200 docs	0.2831
At 500 docs	0.1758
At 1000 docs	0.1019
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3775



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UMassX2
Run Description	English topics, automatic, title+desc
Number of Topics:	50

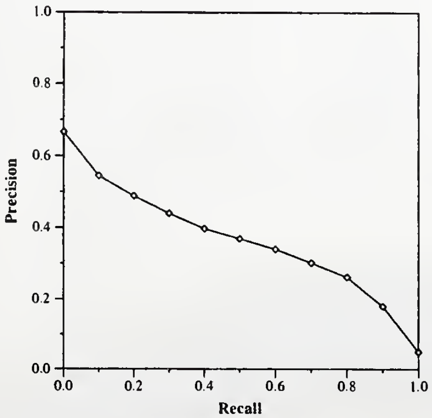
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	5130

Recall Level Averages	
Recall	Precision
0.00	0.6664
0.10	0.5448
0.20	0.4880
0.30	0.4399
0.40	0.3972
0.50	0.3691
0.60	0.3395
0.70	0.3011
0.80	0.2615
0.90	0.1797
1.00	0.0494

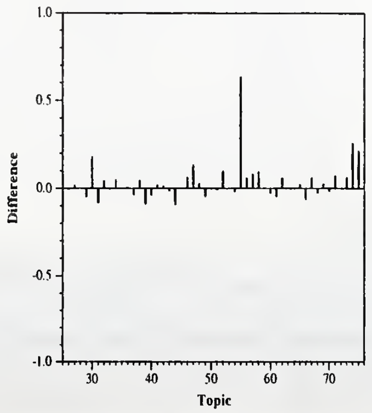
Mean average precision	
non-interpolated	0.3538

Document Level Averages	
	Precision
At 5 docs	0.4640
At 10 docs	0.4420
At 15 docs	0.4120
At 20 docs	0.4080
At 30 docs	0.3920
At 100 docs	0.3302
At 200 docs	0.2755
At 500 docs	0.1740
At 1000 docs	0.1026

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3621



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UMassX2n
Run Description	English topics, automatic, title+desc+narr
Number of Topics:	50

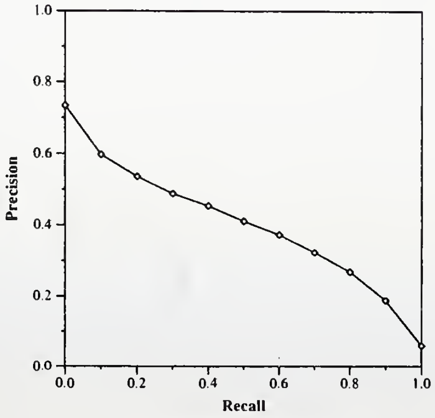
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	5081

Recall Level Averages	
Recall	Precision
0.00	0.7339
0.10	0.5966
0.20	0.5361
0.30	0.4882
0.40	0.4532
0.50	0.4103
0.60	0.3722
0.70	0.3227
0.80	0.2682
0.90	0.1878
1.00	0.0588

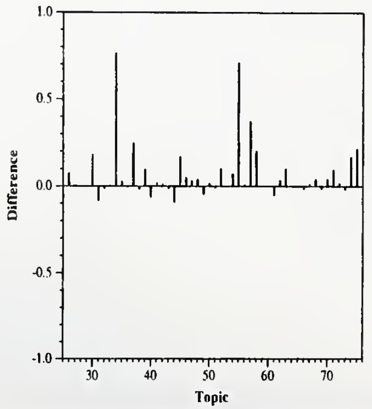
Mean average precision	
non-interpolated	0.3900

Document Level Averages	
	Precision
At 5 docs	0.5240
At 10 docs	0.4740
At 15 docs	0.4493
At 20 docs	0.4350
At 30 docs	0.4133
At 100 docs	0.3388
At 200 docs	0.2779
At 500 docs	0.1742
At 1000 docs	0.1016

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.4020



Recall-Precision Curve



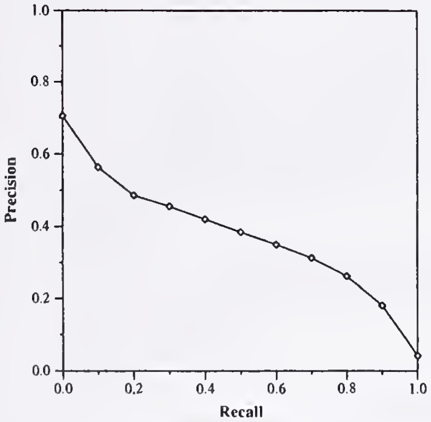
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UMassX6
Run Description	English topics, automatic, title+desc
Number of Topics:	50

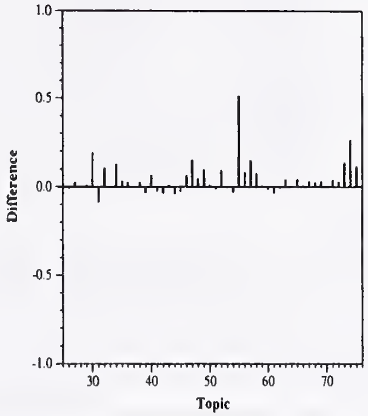
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	5149

Recall Level Averages	
Recall	Precision
0.00	0.7054
0.10	0.5636
0.20	0.4863
0.30	0.4554
0.40	0.4197
0.50	0.3841
0.60	0.3495
0.70	0.3128
0.80	0.2621
0.90	0.1806
1.00	0.0415
Mean average precision	
non-interpolated	0.3658

Document Level Averages	
	Precision
At 5 docs	0.4840
At 10 docs	0.4540
At 15 docs	0.4307
At 20 docs	0.4250
At 30 docs	0.4047
At 100 docs	0.3374
At 200 docs	0.2848
At 500 docs	0.1755
At 1000 docs	0.1030
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3824



Recall-Precision Curve



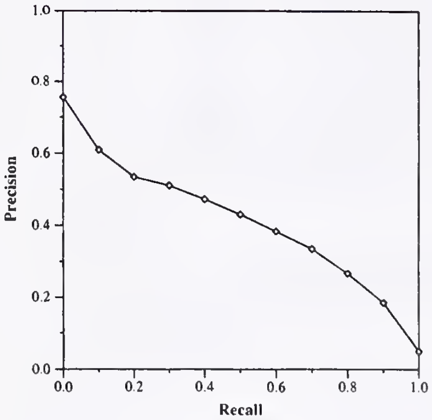
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UMassX6n
Run Description	English topics, automatic, title+desc+narr
Number of Topics:	50

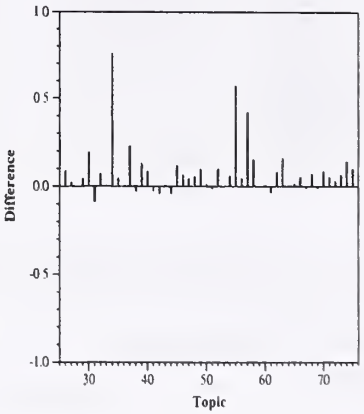
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	5111

Recall Level Averages	
Recall	Precision
0.00	0.7555
0.10	0.6088
0.20	0.5338
0.30	0.5105
0.40	0.4724
0.50	0.4301
0.60	0.3830
0.70	0.3348
0.80	0.2667
0.90	0.1848
1.00	0.0496
Mean average precision	
non-interpolated	0.3996

Document Level Averages	
	Precision
At 5 docs	0.5200
At 10 docs	0.4880
At 15 docs	0.4760
At 20 docs	0.4600
At 30 docs	0.4273
At 100 docs	0.3488
At 200 docs	0.2850
At 500 docs	0.1764
At 1000 docs	0.1022
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.4207



Recall-Precision Curve



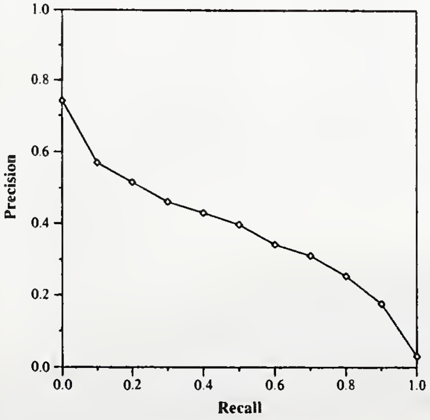
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UniNE1
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	50

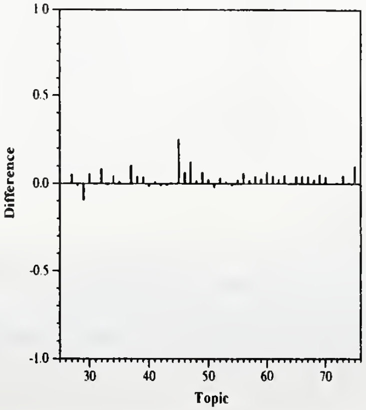
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	5113

Recall Level Averages	
Recall	Precision
0.00	0.7417
0.10	0.5703
0.20	0.5164
0.30	0.4618
0.40	0.4307
0.50	0.3975
0.60	0.3422
0.70	0.3110
0.80	0.2539
0.90	0.1762
1.00	0.0304
Mean average precision	
non-interpolated	0.3712

Document Level Averages	
	Precision
At 5 docs	0.5200
At 10 docs	0.4780
At 15 docs	0.4600
At 20 docs	0.4430
At 30 docs	0.4147
At 100 docs	0.3426
At 200 docs	0.2838
At 500 docs	0.1740
At 1000 docs	0.1023
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3857



Recall-Precision Curve



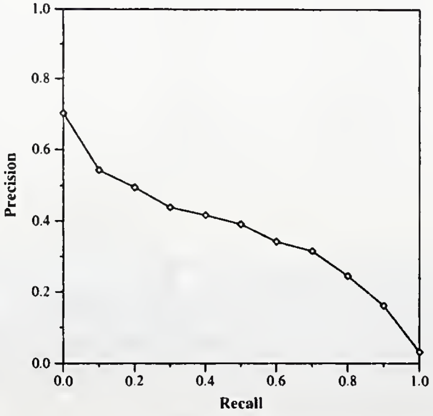
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UniNE2
Run Description	Arabic topics, automatic, title
Number of Topics:	50

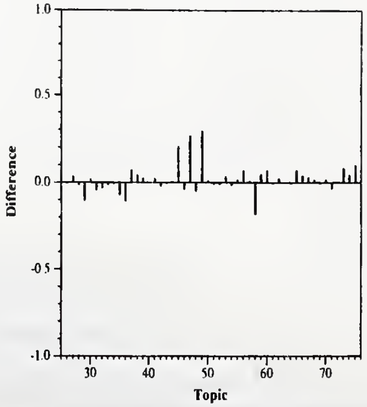
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	5026

Recall Level Averages	
Recall	Precision
0.00	0.7010
0.10	0.5434
0.20	0.4950
0.30	0.4390
0.40	0.4166
0.50	0.3911
0.60	0.3425
0.70	0.3159
0.80	0.2455
0.90	0.1613
1.00	0.0315
Mean average precision	
non-interpolated	0.3572

Document Level Averages	
	Precision
At 5 docs	0.5040
At 10 docs	0.4580
At 15 docs	0.4280
At 20 docs	0.4180
At 30 docs	0.3947
At 100 docs	0.3376
At 200 docs	0.2811
At 500 docs	0.1722
At 1000 docs	0.1005
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3769



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UniNE3
Run Description	Arabic topics, automatic, title+desc+narr
Number of Topics:	50

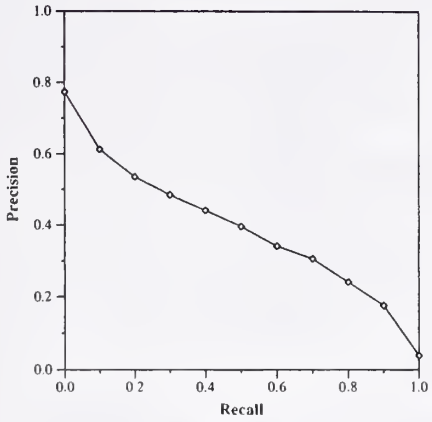
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	5036

Recall Level Averages	
Recall	Precision
0.00	0.7735
0.10	0.6124
0.20	0.5356
0.30	0.4847
0.40	0.4414
0.50	0.3956
0.60	0.3405
0.70	0.3055
0.80	0.2406
0.90	0.1758
1.00	0.0388

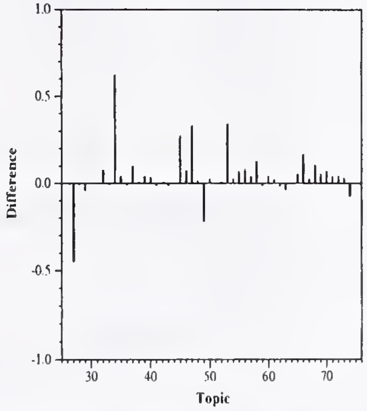
Mean average precision	
non-interpolated	0.3807

Document Level Averages	
	Precision
At 5 docs	0.5760
At 10 docs	0.5160
At 15 docs	0.4800
At 20 docs	0.4630
At 30 docs	0.4293
At 100 docs	0.3388
At 200 docs	0.2772
At 500 docs	0.1725
At 1000 docs	0.1007

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3955



Recall-Precision Curve



Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID:	UniNE4
Run Description	Arabic topics, automatic, title+desc
Number of Topics:	50

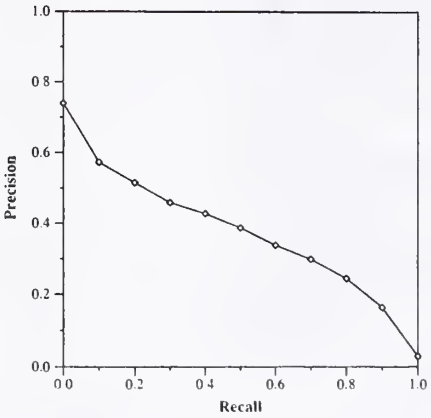
Total number of documents over all topics	
Retrieved:	50000
Relevant:	5909
Rel-ret:	4998

Recall Level Averages	
Recall	Precision
0.00	0.7396
0.10	0.5727
0.20	0.5150
0.30	0.4591
0.40	0.4275
0.50	0.3871
0.60	0.3379
0.70	0.2985
0.80	0.2448
0.90	0.1636
1.00	0.0287

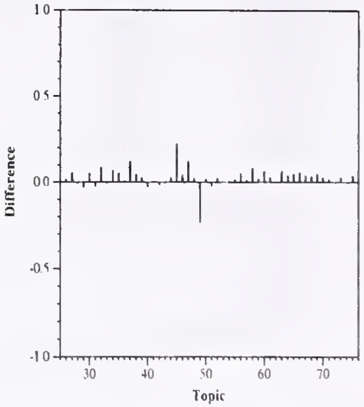
Mean average precision	
non-interpolated	0.3660

Document Level Averages	
	Precision
At 5 docs	0.5120
At 10 docs	0.4820
At 15 docs	0.4520
At 20 docs	0.4480
At 30 docs	0.4153
At 100 docs	0.3328
At 200 docs	0.2795
At 500 docs	0.1699
At 1000 docs	0.1000

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.3869



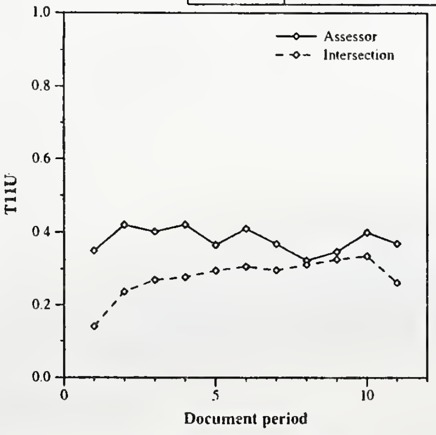
Recall-Precision Curve



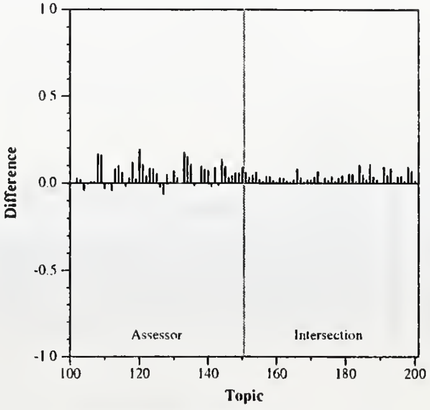
Difference from Median in Average Precision per Topic

Summary Statistics		
Run ID:		CMUDIRFDESC
Subtask		adaptive
Resources used:		none
Optimized for		T11F
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2291	558
Rel-ret:	1557	148
Mean T11U	0.431	0.293
Mean T11F	0.401	0.038
Mean Set Recall	0.269	0.016
Mean Set Precision	0.509	0.063
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.350	0.140
2	158952 - 226929	0.420	0.237
3	226930 - 282799	0.402	0.270
4	282800 - 349081	0.421	0.278
5	349082 - 412613	0.366	0.296
6	412614 - 477886	0.410	0.307
7	477887 - 552759	0.369	0.297
8	552760 - 628903	0.324	0.312
9	628904 - 697981	0.348	0.327
10	697982 - 771567	0.400	0.336
11	771568 - 810597	0.370	0.263



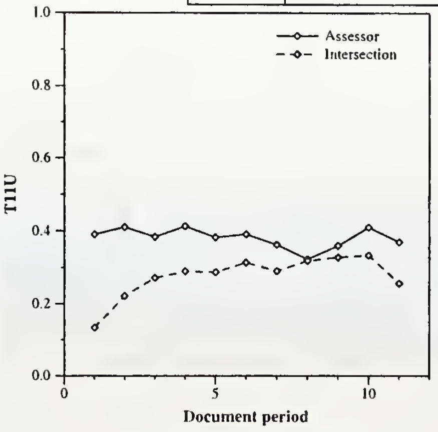
T11U within time periods



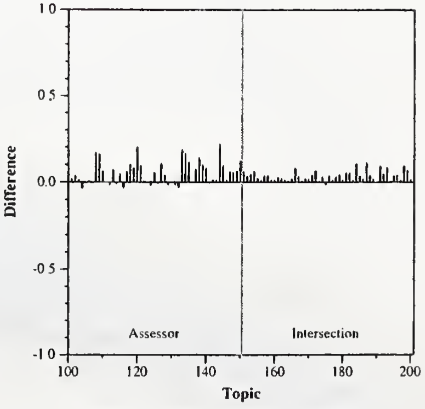
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		CMUDIRFml
Subtask		adaptive
Resources used:		none
Optimized for		T11F
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2338	507
Rel-ret:	1595	118
Mean T11U	0.433	0.292
Mean T11F	0.396	0.035
Mean Set Recall	0.267	0.014
Mean Set Precision	0.499	0.060
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.391	0.133
2	158952 - 226929	0.411	0.222
3	226930 - 282799	0.384	0.272
4	282800 - 349081	0.413	0.290
5	349082 - 412613	0.383	0.288
6	412614 - 477886	0.391	0.314
7	477887 - 552759	0.363	0.291
8	552760 - 628903	0.324	0.319
9	628904 - 697981	0.360	0.329
10	697982 - 771567	0.410	0.334
11	771568 - 810597	0.370	0.257



T11U within time periods

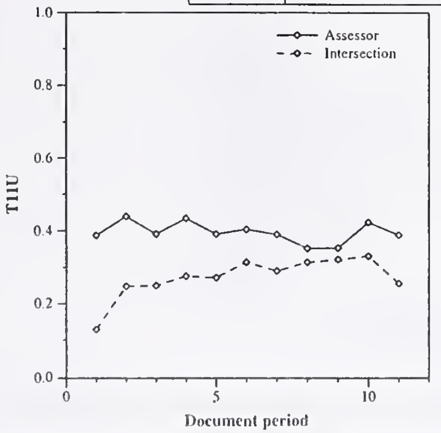


Difference from Median in T11U per Topic

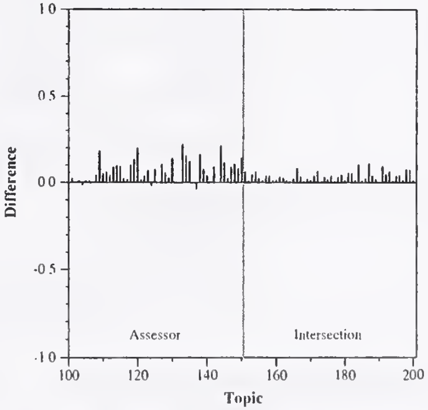
Filtering - Adaptive filtering results — Carnegie Mellon University (CMUDIR)

Summary Statistics		
Run ID:		CMUDIRUDESC
Subtask		adaptive
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2570	567
Rel-ret:	1724	141
Mean T11U	0.445	0.291
Mean T11F	0.422	0.041
Mean Set Recall	0.298	0.017
Mean Set Precision	0.517	0.072
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.387	0.129
2	158952 - 226929	0.439	0.248
3	226930 - 282799	0.391	0.250
4	282800 - 349081	0.434	0.276
5	349082 - 412613	0.391	0.272
6	412614 - 477886	0.404	0.314
7	477887 - 552759	0.390	0.290
8	552760 - 628903	0.352	0.314
9	628904 - 697981	0.353	0.321
10	697982 - 771567	0.423	0.331
11	771568 - 810597	0.388	0.256



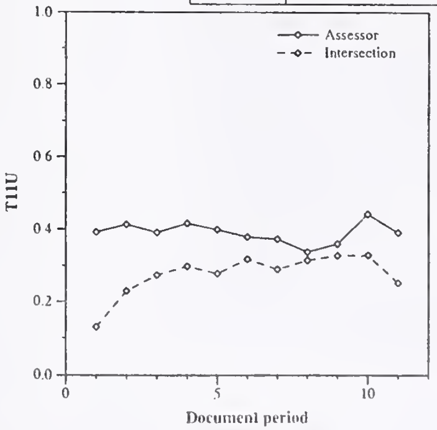
T11U within time periods



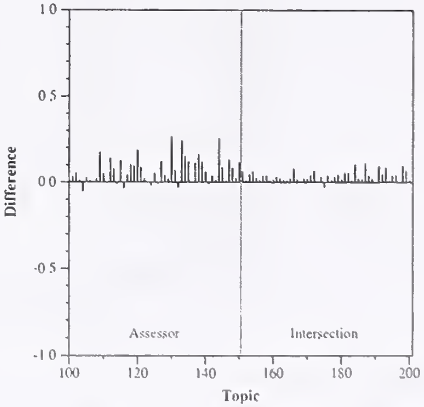
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		CMUDIRUml
Subtask		adaptive
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2580	497
Rel-ret:	1715	102
Mean T11U	0.447	0.290
Mean T11F	0.410	0.034
Mean Set Recall	0.301	0.013
Mean Set Precision	0.497	0.061
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.392	0.130
2	158952 - 226929	0.413	0.230
3	226930 - 282799	0.391	0.274
4	282800 - 349081	0.416	0.298
5	349082 - 412613	0.399	0.278
6	412614 - 477886	0.379	0.318
7	477887 - 552759	0.373	0.290
8	552760 - 628903	0.338	0.315
9	628904 - 697981	0.360	0.328
10	697982 - 771567	0.442	0.329
11	771568 - 810597	0.391	0.253



T11U within time periods

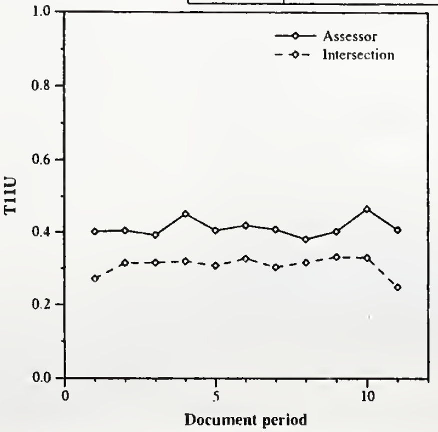


Difference from Median in T11U per Topic

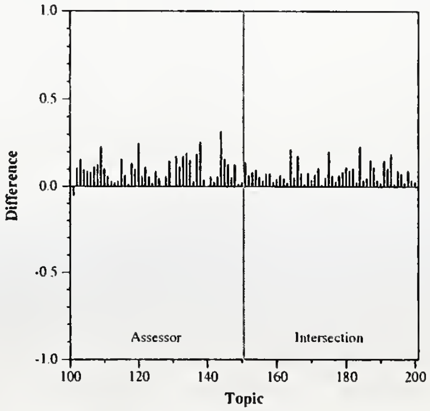
Filtering - Adaptive filtering results — Chinese Academy of Sciences

Summary Statistics		
Run ID:		ICTAdaFT11Ua
Subtask		adaptive
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	3364	802
Rel-ret:	2134	398
Mean T11U	0.475	0.335
Mean T11F	0.427	0.061
Mean Set Recall	0.355	0.039
Mean Set Precision	0.505	0.115
Zero returns	0	4

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.402	0.272
2	158952 - 226929	0.405	0.315
3	226930 - 282799	0.393	0.316
4	282800 - 349081	0.452	0.320
5	349082 - 412613	0.406	0.308
6	412614 - 477886	0.420	0.328
7	477887 - 552759	0.409	0.304
8	552760 - 628903	0.382	0.318
9	628904 - 697981	0.404	0.333
10	697982 - 771567	0.467	0.331
11	771568 - 810597	0.409	0.250



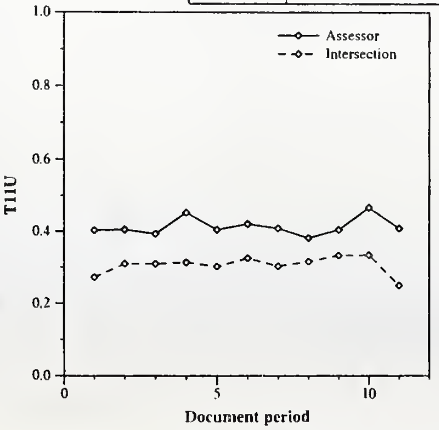
T11U within time periods



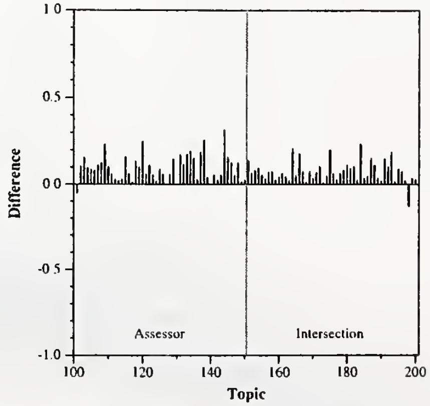
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		ICTAdaFT11Ub
Subtask		adaptive
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	3372	889
Rel-ret:	2143	400
Mean T11U	0.475	0.330
Mean T11F	0.428	0.062
Mean Set Recall	0.356	0.041
Mean Set Precision	0.506	0.115
Zero returns	0	4

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.403	0.272
2	158952 - 226929	0.405	0.309
3	226930 - 282799	0.393	0.309
4	282800 - 349081	0.453	0.313
5	349082 - 412613	0.405	0.302
6	412614 - 477886	0.421	0.325
7	477887 - 552759	0.409	0.303
8	552760 - 628903	0.382	0.316
9	628904 - 697981	0.405	0.333
10	697982 - 771567	0.467	0.334
11	771568 - 810597	0.409	0.250



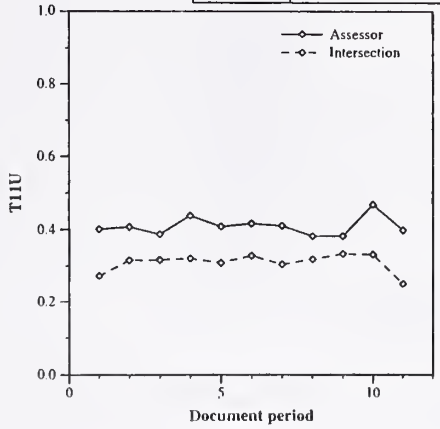
T11U within time periods



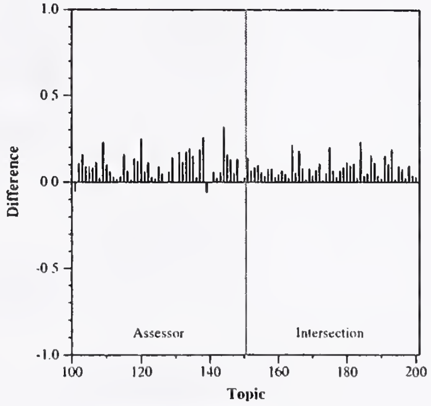
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		ICTAdaFT11Uc
Subtask		adaptive
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	3412	802
Rel-ret:	2148	398
Mean T11U	0.471	0.335
Mean T11F	0.422	0.061
Mean Set Recall	0.361	0.039
Mean Set Precision	0.498	0.115
Zero returns	0	4

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.401	0.272
2	158952 - 226929	0.407	0.315
3	226930 - 282799	0.387	0.316
4	282800 - 349081	0.438	0.320
5	349082 - 412613	0.408	0.308
6	412614 - 477886	0.416	0.328
7	477887 - 552759	0.410	0.304
8	552760 - 628903	0.382	0.318
9	628904 - 697981	0.382	0.333
10	697982 - 771567	0.468	0.331
11	771568 - 810597	0.398	0.250



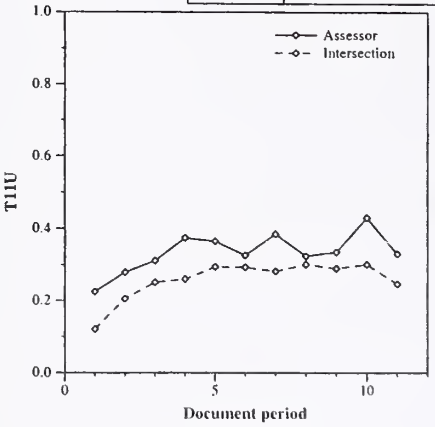
T11U within time periods



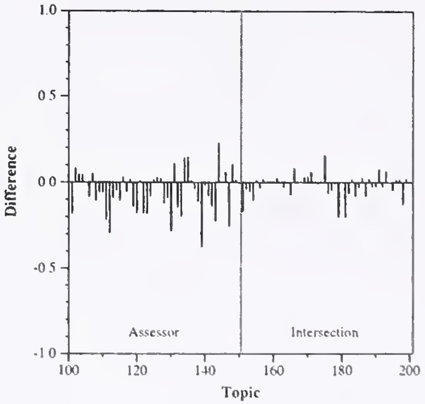
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		ICTAdaFT11Ud
Subtask		adaptive
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	4302	1388
Rel-ret:	1958	340
Mean T11U	0.321	0.240
Mean T11F	0.306	0.052
Mean Set Recall	0.350	0.038
Mean Set Precision	0.321	0.063
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.225	0.120
2	158952 - 226929	0.279	0.205
3	226930 - 282799	0.311	0.251
4	282800 - 349081	0.374	0.260
5	349082 - 412613	0.365	0.294
6	412614 - 477886	0.326	0.293
7	477887 - 552759	0.385	0.282
8	552760 - 628903	0.324	0.301
9	628904 - 697981	0.335	0.290
10	697982 - 771567	0.430	0.301
11	771568 - 810597	0.330	0.247



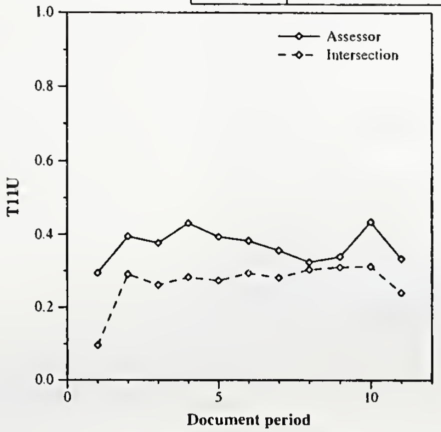
T11U within time periods



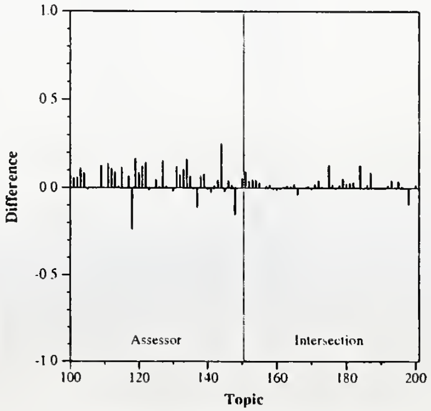
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		reliefst11u
Subtask		adaptive
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2489	746
Rel-ret:	1565	191
Mean T11U	0.424	0.274
Mean T11F	0.369	0.033
Mean Set Recall	0.282	0.018
Mean Set Precision	0.443	0.052
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.294	0.096
2	158952 - 226929	0.395	0.291
3	226930 - 282799	0.377	0.262
4	282800 - 349081	0.431	0.283
5	349082 - 412613	0.394	0.274
6	412614 - 477886	0.383	0.294
7	477887 - 552759	0.357	0.282
8	552760 - 628903	0.325	0.304
9	628904 - 697981	0.340	0.311
10	697982 - 771567	0.435	0.313
11	771568 - 810597	0.334	0.241



T11U within time periods

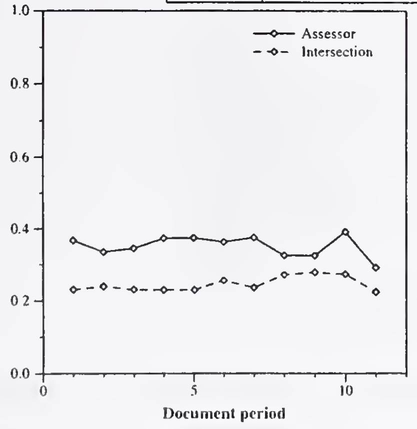


Difference from Median in T11U per Topic

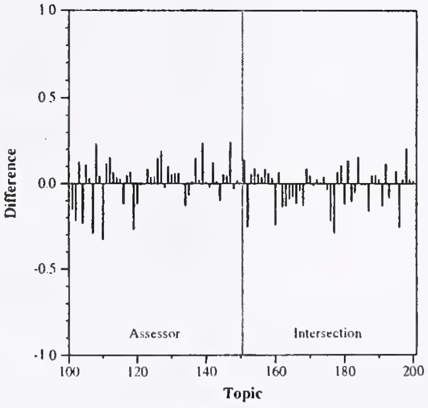
Summary Statistics		
Run ID:		FDUT11AF1
Subtask		adaptive
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2280	1883
Rel-ret:	1065	196
Mean T11U	0.393	0.240
Mean T11F	0.329	0.047
Mean Set Recall	0.191	0.040
Mean Set Precision	0.563	0.066
Zero returns	6	10

T11U within time periods

Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.368	0.231
2	158952 - 226929	0.336	0.239
3	226930 - 282799	0.346	0.231
4	282800 - 349081	0.374	0.230
5	349082 - 412613	0.375	0.230
6	412614 - 477886	0.364	0.256
7	477887 - 552759	0.376	0.236
8	552760 - 628903	0.326	0.272
9	628904 - 697981	0.326	0.279
10	697982 - 771567	0.392	0.274
11	771568 - 810597	0.292	0.224



T11U within time periods

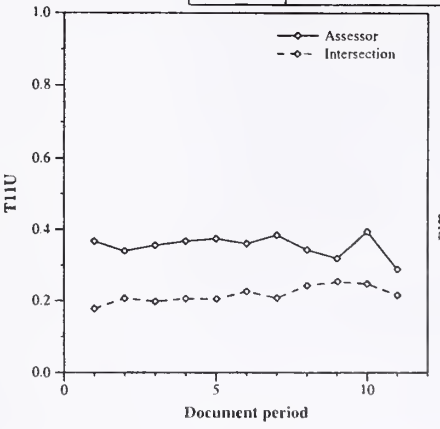


Difference from Median in T11U per Topic

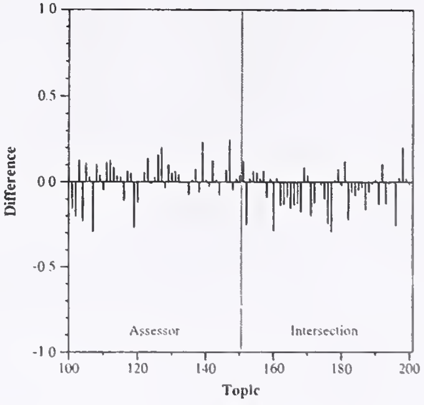
Summary Statistics		
Run ID:		FDUT11AF2
Subtask		adaptive
Resources used:		none
Optimized for		T11F
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2239	3107
Rel-ret:	1143	162
Mean T11U	0.397	0.203
Mean T11F	0.346	0.042
Mean Set Recall	0.212	0.039
Mean Set Precision	0.548	0.056
Zero returns	4	3

T11U within time periods

Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.367	0.178
2	158952 - 226929	0.340	0.207
3	226930 - 282799	0.356	0.198
4	282800 - 349081	0.368	0.206
5	349082 - 412613	0.375	0.205
6	412614 - 477886	0.361	0.226
7	477887 - 552759	0.385	0.208
8	552760 - 628903	0.344	0.243
9	628904 - 697981	0.320	0.254
10	697982 - 771567	0.395	0.248
11	771568 - 810597	0.289	0.216



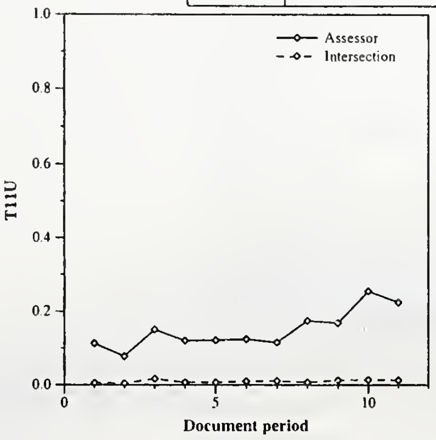
T11U within time periods



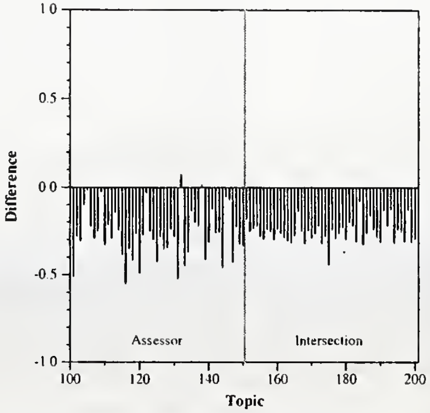
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		dimacsddl02a
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	15143	299040
Rel-ret:	1868	1175
Mean T11U	0.110	0.002
Mean T11F	0.125	0.008
Mean Set Recall	0.233	0.161
Mean Set Precision	0.145	0.007
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.113	0.005
2	158952 - 226929	0.078	0.004
3	226930 - 282799	0.151	0.017
4	282800 - 349081	0.121	0.007
5	349082 - 412613	0.122	0.007
6	412614 - 477886	0.125	0.010
7	477887 - 552759	0.116	0.011
8	552760 - 628903	0.175	0.007
9	628904 - 697981	0.169	0.013
10	697982 - 771567	0.255	0.014
11	771568 - 810597	0.225	0.013



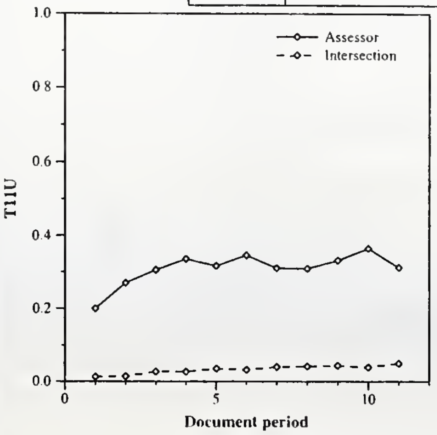
T11U within time periods



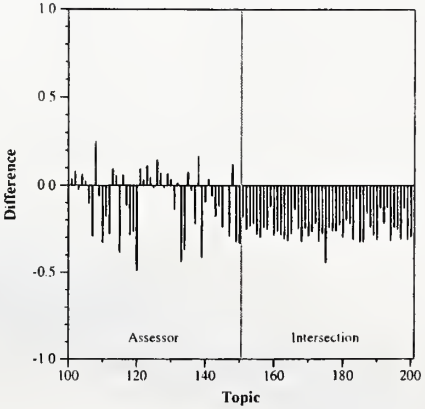
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		dimacsddl02b
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11F
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	6556	366113
Rel-ret:	2420	1784
Mean T11U	0.293	0.004
Mean T11F	0.318	0.023
Mean Set Recall	0.358	0.291
Mean Set Precision	0.337	0.020
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.200	0.013
2	158952 - 226929	0.270	0.014
3	226930 - 282799	0.305	0.027
4	282800 - 349081	0.335	0.027
5	349082 - 412613	0.316	0.035
6	412614 - 477886	0.345	0.032
7	477887 - 552759	0.310	0.040
8	552760 - 628903	0.309	0.042
9	628904 - 697981	0.331	0.044
10	697982 - 771567	0.364	0.039
11	771568 - 810597	0.312	0.050



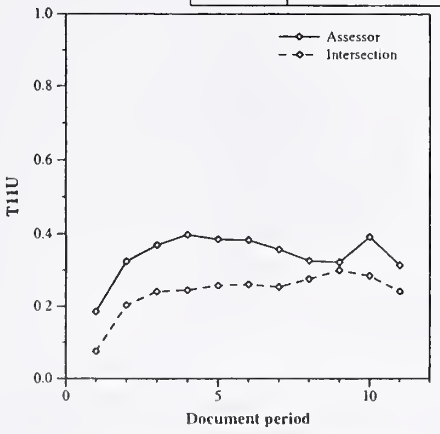
T11U within time periods



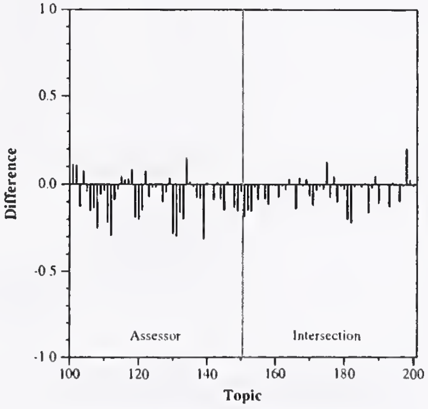
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		iritsigal
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	3030	1426
Rel-ret:	1396	272
Mean T11U	0.312	0.219
Mean T11F	0.285	0.053
Mean Set Recall	0.281	0.034
Mean Set Precision	0.332	0.083
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.185	0.075
2	158952 - 226929	0.324	0.203
3	226930 - 282799	0.369	0.241
4	282800 - 349081	0.398	0.245
5	349082 - 412613	0.385	0.258
6	412614 - 477886	0.383	0.261
7	477887 - 552759	0.357	0.254
8	552760 - 628903	0.326	0.276
9	628904 - 697981	0.322	0.300
10	697982 - 771567	0.392	0.285
11	771568 - 810597	0.314	0.242



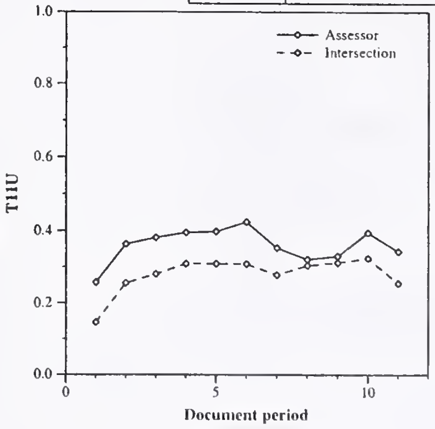
T11U within time periods



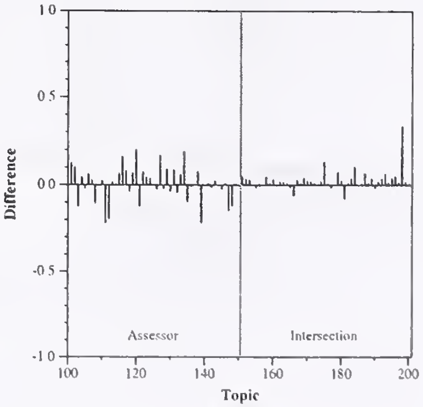
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		iritsiga2
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2237	913
Rel-ret:	1325	266
Mean T11U	0.386	0.282
Mean T11F	0.327	0.054
Mean Set Recall	0.261	0.031
Mean Set Precision	0.410	0.092
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.257	0.146
2	158952 - 226929	0.363	0.255
3	226930 - 282799	0.381	0.280
4	282800 - 349081	0.395	0.309
5	349082 - 412613	0.398	0.309
6	412614 - 477886	0.424	0.308
7	477887 - 552759	0.353	0.278
8	552760 - 628903	0.321	0.304
9	628904 - 697981	0.330	0.312
10	697982 - 771567	0.395	0.324
11	771568 - 810597	0.343	0.255



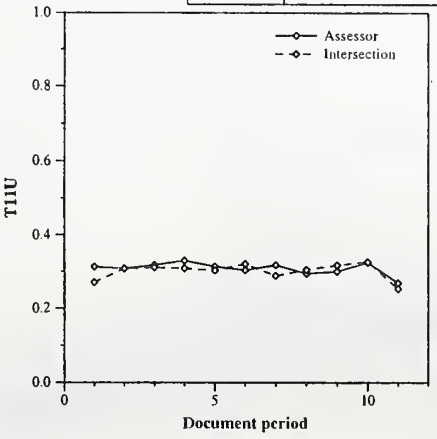
T11U within time periods



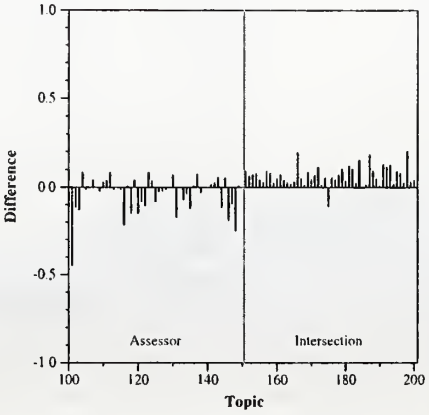
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		apl11Fah1
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	325	91
Rel-ret:	200	14
Mean T11U	0.342	0.322
Mean T11F	0.104	0.014
Mean Set Recall	0.039	0.003
Mean Set Precision	0.377	0.064
Zero returns	11	26

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.312	0.270
2	158952 - 226929	0.307	0.307
3	226930 - 282799	0.317	0.310
4	282800 - 349081	0.329	0.308
5	349082 - 412613	0.313	0.302
6	412614 - 477886	0.303	0.320
7	477887 - 552759	0.317	0.288
8	552760 - 628903	0.294	0.304
9	628904 - 697981	0.300	0.317
10	697982 - 771567	0.324	0.326
11	771568 - 810597	0.269	0.253



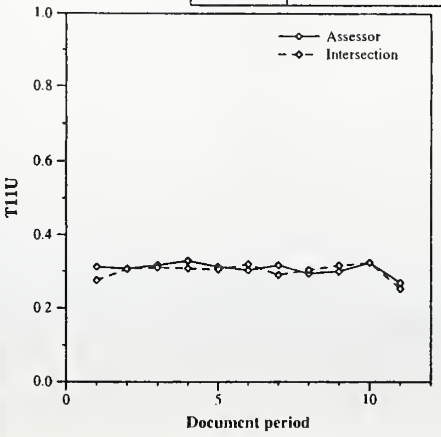
T11U within time periods



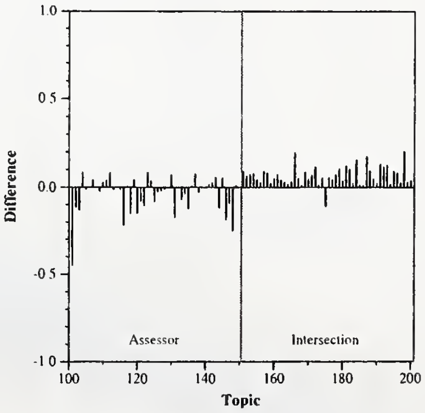
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		apl11Fah2
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	325	71
Rel-ret:	200	9
Mean T11U	0.342	0.322
Mean T11F	0.104	0.009
Mean Set Recall	0.039	0.002
Mean Set Precision	0.377	0.051
Zero returns	11	28

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.312	0.275
2	158952 - 226929	0.307	0.306
3	226930 - 282799	0.317	0.310
4	282800 - 349081	0.329	0.308
5	349082 - 412613	0.313	0.305
6	412614 - 477886	0.303	0.320
7	477887 - 552759	0.317	0.290
8	552760 - 628903	0.294	0.304
9	628904 - 697981	0.300	0.316
10	697982 - 771567	0.324	0.324
11	771568 - 810597	0.269	0.253



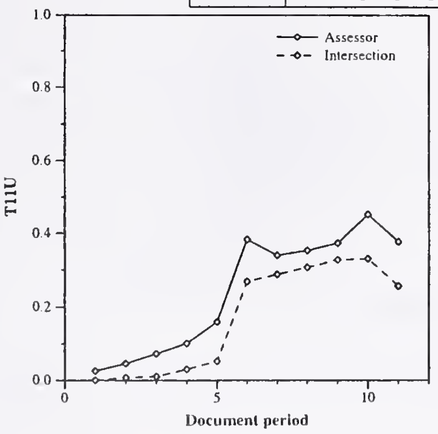
T11U within time periods



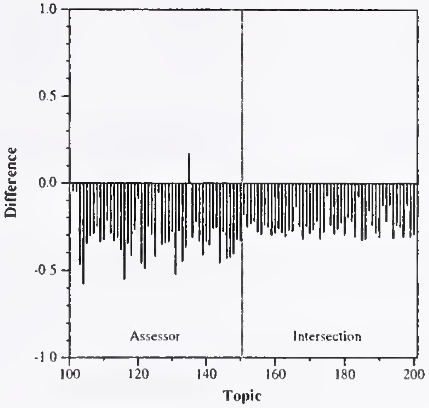
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		apl11Faq1
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	45142	49398
Rel-ret:	2114	604
Mean T11U	0.059	0.008
Mean T11F	0.090	0.021
Mean Set Recall	0.369	0.115
Mean Set Precision	0.084	0.018
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.026	0.000
2	158952 - 226929	0.047	0.008
3	226930 - 282799	0.074	0.011
4	282800 - 349081	0.102	0.031
5	349082 - 412613	0.160	0.053
6	412614 - 477886	0.383	0.269
7	477887 - 552759	0.340	0.288
8	552760 - 628903	0.353	0.307
9	628904 - 697981	0.373	0.327
10	697982 - 771567	0.452	0.331
11	771568 - 810597	0.377	0.257



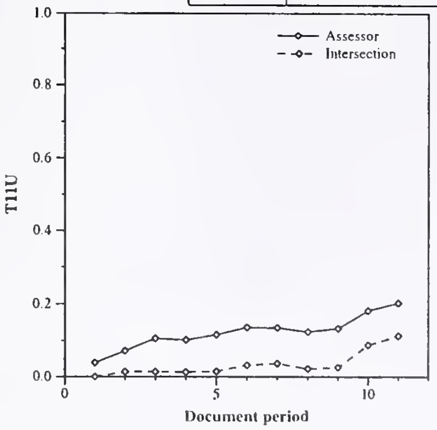
T11U within time periods



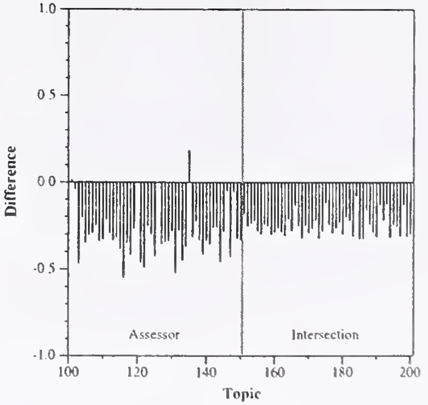
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		apl11Faq2
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	32742	41924
Rel-ret:	2049	532
Mean T11U	0.085	0.009
Mean T11F	0.118	0.022
Mean Set Recall	0.355	0.110
Mean Set Precision	0.115	0.020
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.039	0.000
2	158952 - 226929	0.072	0.014
3	226930 - 282799	0.106	0.014
4	282800 - 349081	0.102	0.013
5	349082 - 412613	0.116	0.015
6	412614 - 477886	0.136	0.032
7	477887 - 552759	0.135	0.037
8	552760 - 628903	0.124	0.022
9	628904 - 697981	0.133	0.026
10	697982 - 771567	0.182	0.088
11	771568 - 810597	0.203	0.113



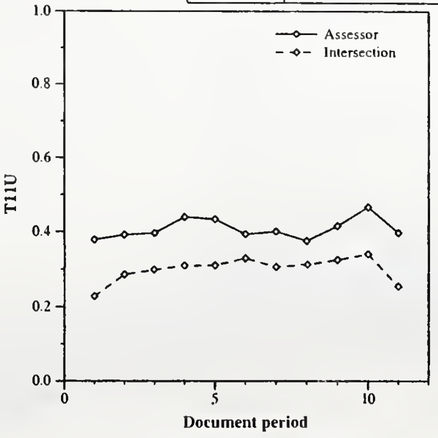
T11U within time periods



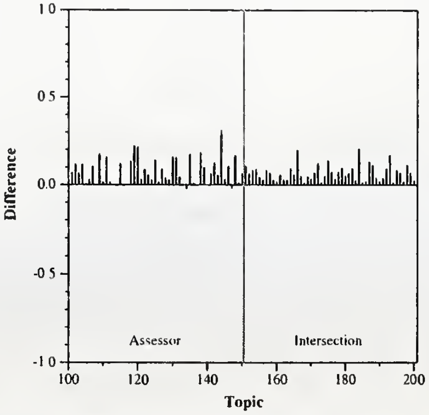
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		KerMITT11af1
Subtask		adaptive
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2433	443
Rel-ret:	1795	215
Mean T11U	0.456	0.323
Mean T11F	0.378	0.049
Mean Set Recall	0.259	0.021
Mean Set Precision	0.545	0.122
Zero returns	0	5

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.378	0.228
2	158952 - 226929	0.391	0.286
3	226930 - 282799	0.396	0.299
4	282800 - 349081	0.439	0.309
5	349082 - 412613	0.433	0.310
6	412614 - 477886	0.393	0.329
7	477887 - 552759	0.400	0.306
8	552760 - 628903	0.375	0.312
9	628904 - 697981	0.415	0.325
10	697982 - 771567	0.466	0.340
11	771568 - 810597	0.397	0.255



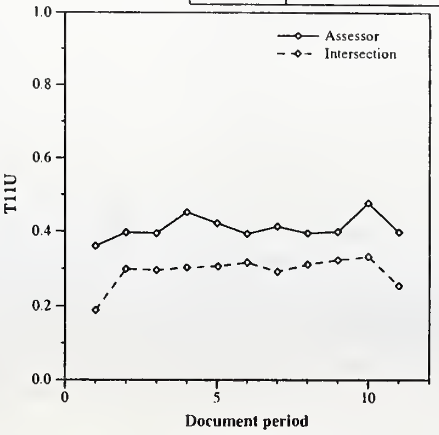
T11U within time periods



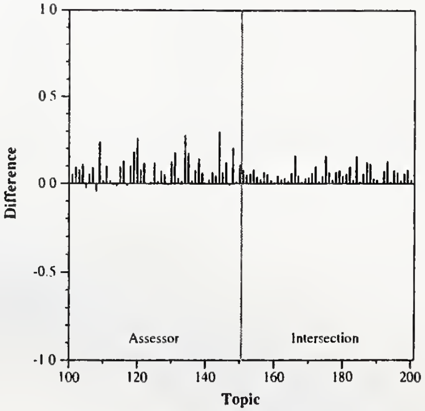
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		KerMITT11af2
Subtask		adaptive
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2799	559
Rel-ret:	1954	232
Mean T11U	0.459	0.311
Mean T11F	0.376	0.047
Mean Set Recall	0.297	0.022
Mean Set Precision	0.473	0.105
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.360	0.189
2	158952 - 226929	0.397	0.299
3	226930 - 282799	0.395	0.296
4	282800 - 349081	0.452	0.303
5	349082 - 412613	0.422	0.306
6	412614 - 477886	0.393	0.316
7	477887 - 552759	0.413	0.292
8	552760 - 628903	0.395	0.311
9	628904 - 697981	0.399	0.323
10	697982 - 771567	0.477	0.332
11	771568 - 810597	0.398	0.255



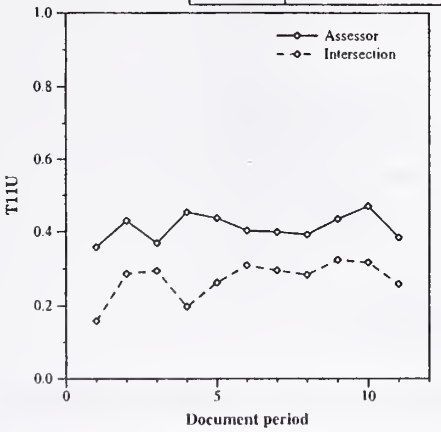
T11U within time periods



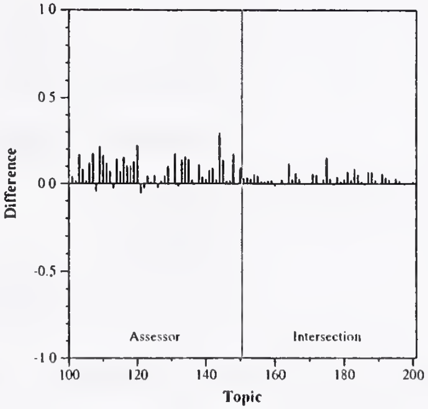
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		KerMITT11af3
Subtask		adaptive
Resources used:		none
Optimized for		T11F
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2504	777
Rel-ret:	1776	250
Mean T11U	0.458	0.285
Mean T11F	0.426	0.048
Mean Set Recall	0.302	0.026
Mean Set Precision	0.527	0.076
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.358	0.159
2	158952 - 226929	0.430	0.287
3	226930 - 282799	0.369	0.295
4	282800 - 349081	0.454	0.198
5	349082 - 412613	0.437	0.263
6	412614 - 477886	0.403	0.309
7	477887 - 552759	0.399	0.296
8	552760 - 628903	0.392	0.284
9	628904 - 697981	0.435	0.324
10	697982 - 771567	0.470	0.317
11	771568 - 810597	0.384	0.258



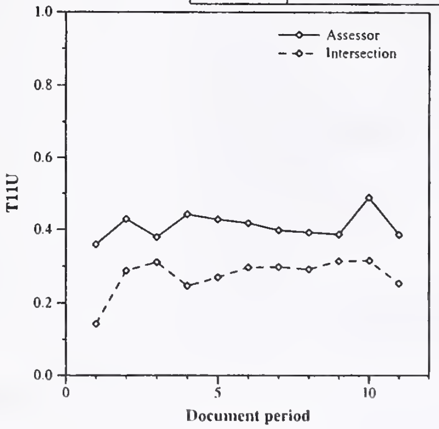
T11U within time periods



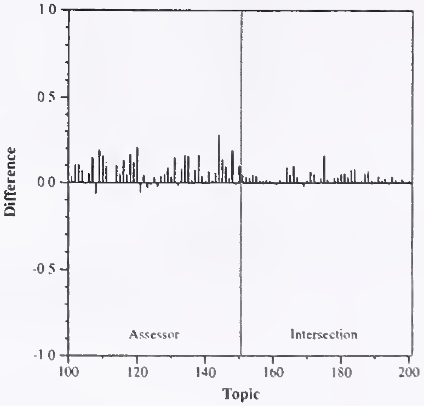
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		KerMITT11af4
Subtask		adaptive
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2668	816
Rel-ret:	1844	268
Mean T11U	0.454	0.287
Mean T11F	0.409	0.056
Mean Set Recall	0.304	0.029
Mean Set Precision	0.506	0.097
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.359	0.143
2	158952 - 226929	0.429	0.288
3	226930 - 282799	0.379	0.311
4	282800 - 349081	0.442	0.247
5	349082 - 412613	0.428	0.270
6	412614 - 477886	0.418	0.297
7	477887 - 552759	0.398	0.298
8	552760 - 628903	0.392	0.292
9	628904 - 697981	0.387	0.314
10	697982 - 771567	0.489	0.315
11	771568 - 810597	0.386	0.254



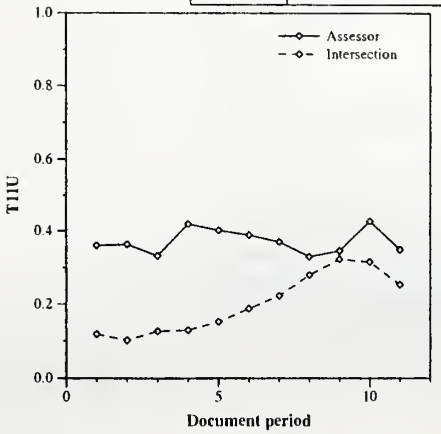
T11U within time periods



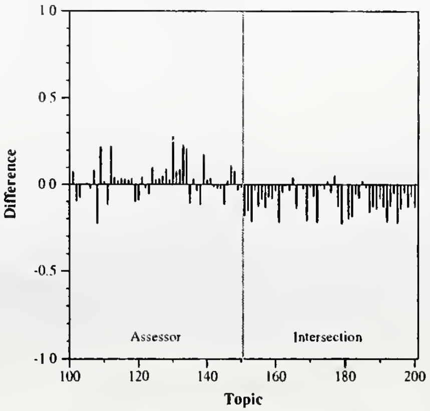
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		ok11aflb
Subtask		adaptive
Resources used:		none
Optimized for		T11F
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	1962	1648
Rel-ret:	1262	158
Mean T11U	0.406	0.163
Mean T11F	0.394	0.043
Mean Set Recall	0.260	0.024
Mean Set Precision	0.529	0.064
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.360	0.118
2	158952 - 226929	0.363	0.102
3	226930 - 282799	0.332	0.126
4	282800 - 349081	0.420	0.129
5	349082 - 412613	0.402	0.152
6	412614 - 477886	0.389	0.188
7	477887 - 552759	0.370	0.223
8	552760 - 628903	0.330	0.280
9	628904 - 697981	0.346	0.323
10	697982 - 771567	0.428	0.316
11	771568 - 810597	0.349	0.254



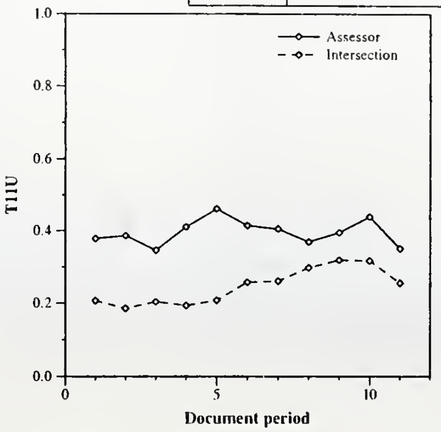
T11U within time periods



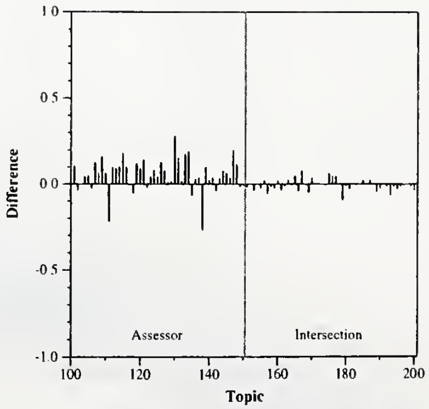
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		ok11aflu
Subtask		adaptive
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2834	979
Rel-ret:	1768	202
Mean T11U	0.435	0.251
Mean T11F	0.421	0.040
Mean Set Recall	0.344	0.023
Mean Set Precision	0.499	0.066
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.378	0.207
2	158952 - 226929	0.386	0.186
3	226930 - 282799	0.345	0.204
4	282800 - 349081	0.410	0.194
5	349082 - 412613	0.461	0.208
6	412614 - 477886	0.414	0.258
7	477887 - 552759	0.405	0.261
8	552760 - 628903	0.369	0.298
9	628904 - 697981	0.395	0.319
10	697982 - 771567	0.439	0.317
11	771568 - 810597	0.350	0.256



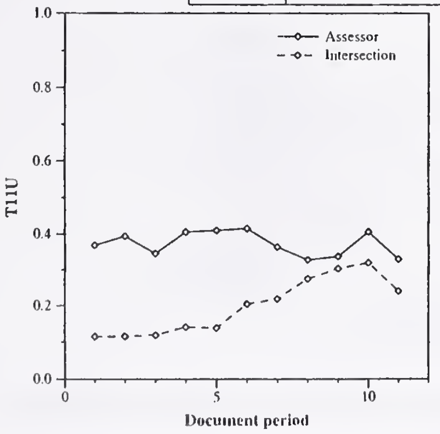
T11U within time periods



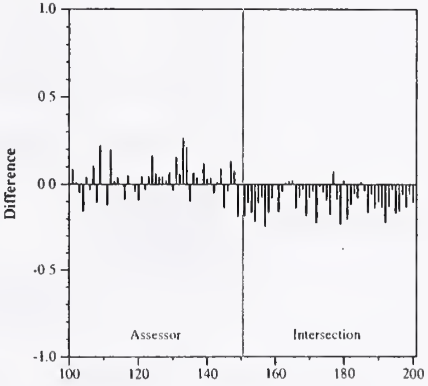
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		ok11afsb
Subtask		adaptive
Resources used:		none
Optimized for		T11F
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2035	1470
Rel-ret:	1306	74
Mean T11U	0.406	0.160
Mean T11F	0.396	0.034
Mean Set Recall	0.259	0.016
Mean Set Precision	0.526	0.062
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.369	0.115
2	158952 - 226929	0.394	0.116
3	226930 - 282799	0.346	0.119
4	282800 - 349081	0.406	0.141
5	349082 - 412613	0.410	0.138
6	412614 - 477886	0.415	0.205
7	477887 - 552759	0.364	0.220
8	552760 - 628903	0.328	0.275
9	628904 - 697981	0.338	0.304
10	697982 - 771567	0.406	0.320
11	771568 - 810597	0.330	0.241



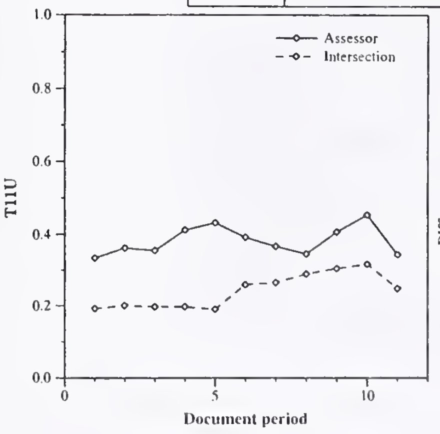
T11U within time periods



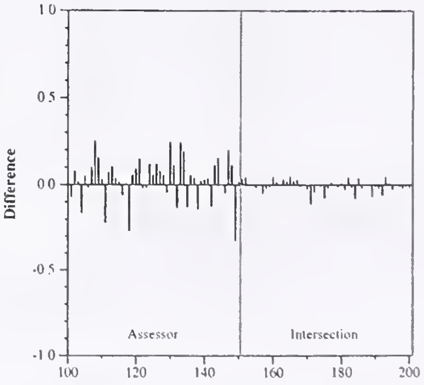
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		ok11afsu
Subtask		adaptive
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2751	778
Rel-ret:	1575	90
Mean T11U	0.409	0.250
Mean T11F	0.403	0.035
Mean Set Recall	0.328	0.017
Mean Set Precision	0.482	0.060
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.334	0.192
2	158952 - 226929	0.362	0.200
3	226930 - 282799	0.355	0.197
4	282800 - 349081	0.412	0.197
5	349082 - 412613	0.432	0.190
6	412614 - 477886	0.392	0.259
7	477887 - 552759	0.367	0.265
8	552760 - 628903	0.346	0.289
9	628904 - 697981	0.407	0.305
10	697982 - 771567	0.454	0.317
11	771568 - 810597	0.344	0.249



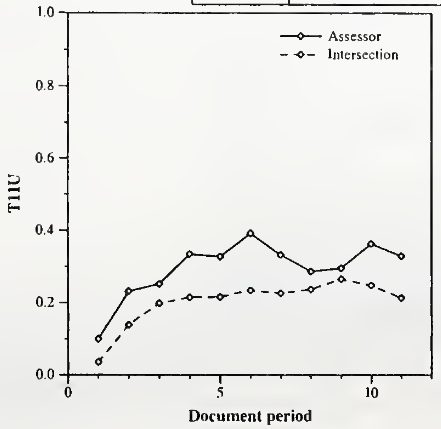
T11U within time periods



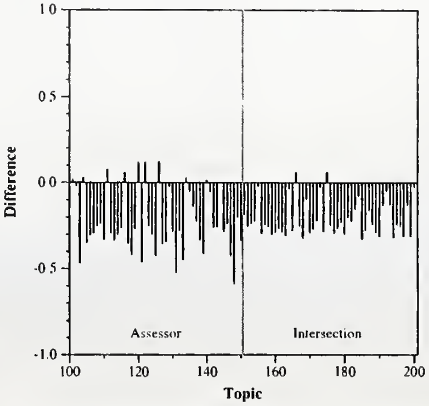
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		pirc2F01
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	13289	14578
Rel-ret:	1816	263
Mean T11U	0.154	0.047
Mean T11F	0.196	0.026
Mean Set Recall	0.379	0.046
Mean Set Precision	0.200	0.028
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.100	0.036
2	158952 - 226929	0.231	0.139
3	226930 - 282799	0.252	0.199
4	282800 - 349081	0.335	0.215
5	349082 - 412613	0.328	0.216
6	412614 - 477886	0.392	0.234
7	477887 - 552759	0.333	0.226
8	552760 - 628903	0.287	0.237
9	628904 - 697981	0.296	0.266
10	697982 - 771567	0.364	0.248
11	771568 - 810597	0.330	0.213



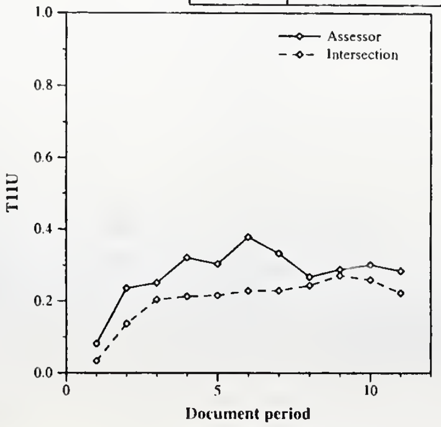
T11U within time periods



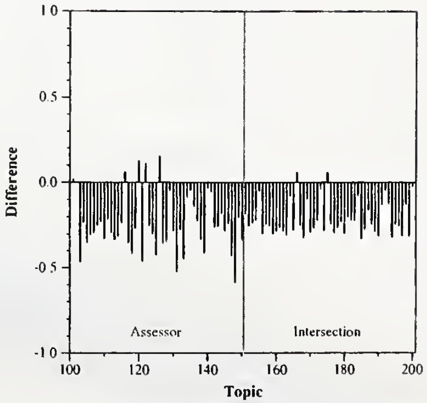
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		pirc2F02
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	13682	15009
Rel-ret:	1857	270
Mean T11U	0.136	0.044
Mean T11F	0.185	0.025
Mean Set Recall	0.393	0.048
Mean Set Precision	0.189	0.027
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.082	0.034
2	158952 - 226929	0.236	0.137
3	226930 - 282799	0.251	0.204
4	282800 - 349081	0.322	0.213
5	349082 - 412613	0.304	0.216
6	412614 - 477886	0.379	0.229
7	477887 - 552759	0.334	0.230
8	552760 - 628903	0.268	0.244
9	628904 - 697981	0.289	0.272
10	697982 - 771567	0.302	0.260
11	771568 - 810597	0.285	0.223



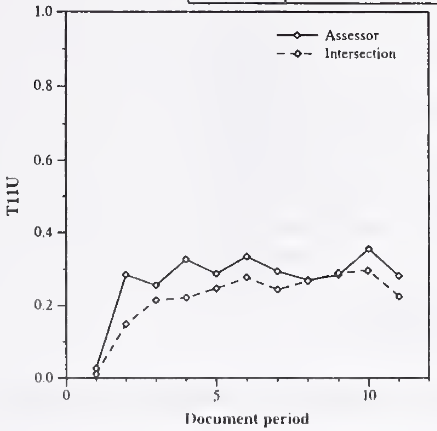
T11U within time periods



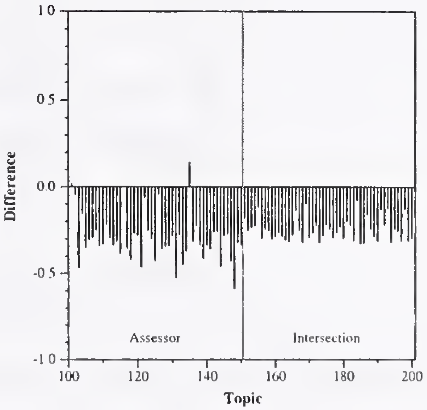
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		pirc2F03
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	18767	19243
Rel-ret:	1818	402
Mean T11U	0.083	0.011
Mean T11F	0.120	0.015
Mean Set Recall	0.294	0.055
Mean Set Precision	0.112	0.013
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.026	0.011
2	158952 - 226929	0.285	0.149
3	226930 - 282799	0.256	0.215
4	282800 - 349081	0.327	0.222
5	349082 - 412613	0.288	0.247
6	412614 - 477886	0.335	0.277
7	477887 - 552759	0.295	0.244
8	552760 - 628903	0.271	0.268
9	628904 - 697981	0.285	0.291
10	697982 - 771567	0.356	0.297
11	771568 - 810597	0.282	0.225



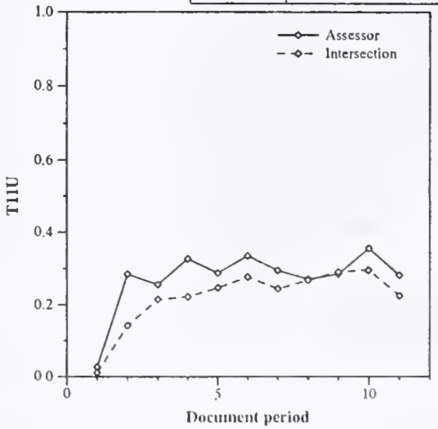
T11U within time periods



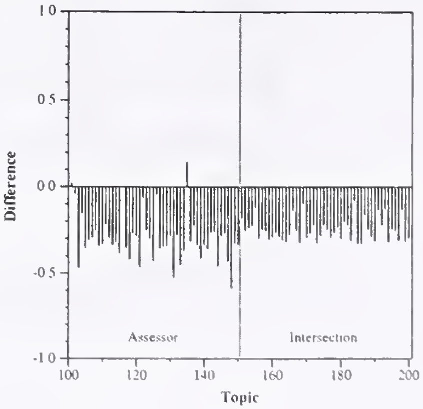
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		pirc2F04
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	18720	19281
Rel-ret:	1815	401
Mean T11U	0.084	0.011
Mean T11F	0.120	0.015
Mean Set Recall	0.293	0.055
Mean Set Precision	0.112	0.013
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.027	0.011
2	158952 - 226929	0.285	0.142
3	226930 - 282799	0.256	0.215
4	282800 - 349081	0.327	0.222
5	349082 - 412613	0.288	0.247
6	412614 - 477886	0.335	0.277
7	477887 - 552759	0.295	0.244
8	552760 - 628903	0.271	0.268
9	628904 - 697981	0.285	0.291
10	697982 - 771567	0.356	0.297
11	771568 - 810597	0.282	0.225



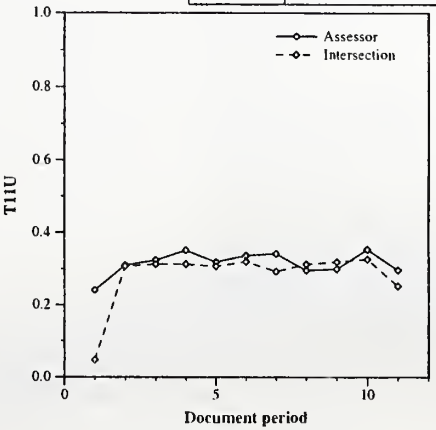
T11U within time periods



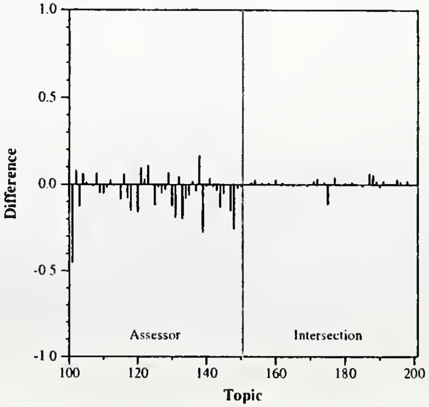
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		dimacs11a30Q
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	1460	500
Rel-ret:	697	15
Mean T11U	0.337	0.262
Mean T11F	0.187	0.013
Mean Set Recall	0.124	0.004
Mean Set Precision	0.278	0.030
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.241	0.047
2	158952 - 226929	0.310	0.307
3	226930 - 282799	0.324	0.313
4	282800 - 349081	0.351	0.313
5	349082 - 412613	0.319	0.307
6	412614 - 477886	0.337	0.320
7	477887 - 552759	0.342	0.293
8	552760 - 628903	0.296	0.313
9	628904 - 697981	0.300	0.320
10	697982 - 771567	0.354	0.327
11	771568 - 810597	0.297	0.253



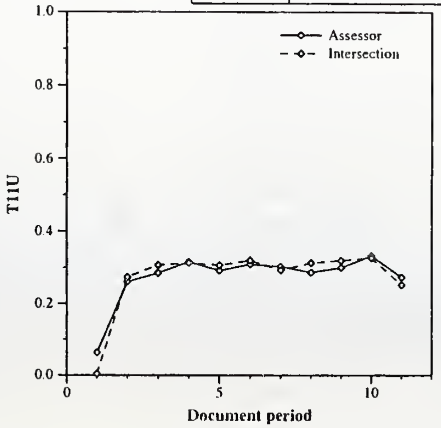
T11U within time periods



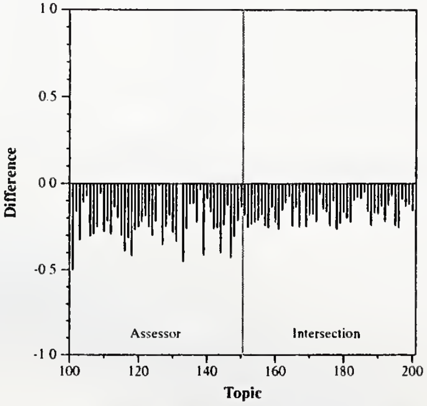
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		dimacs11aAPQ
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2447	2500
Rel-ret:	250	16
Mean T11U	0.142	0.085
Mean T11F	0.068	0.006
Mean Set Recall	0.041	0.006
Mean Set Precision	0.098	0.006
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.063	0.004
2	158952 - 226929	0.261	0.274
3	226930 - 282799	0.285	0.307
4	282800 - 349081	0.316	0.313
5	349082 - 412613	0.292	0.307
6	412614 - 477886	0.309	0.320
7	477887 - 552759	0.303	0.293
8	552760 - 628903	0.287	0.313
9	628904 - 697981	0.300	0.320
10	697982 - 771567	0.333	0.327
11	771568 - 810597	0.274	0.253



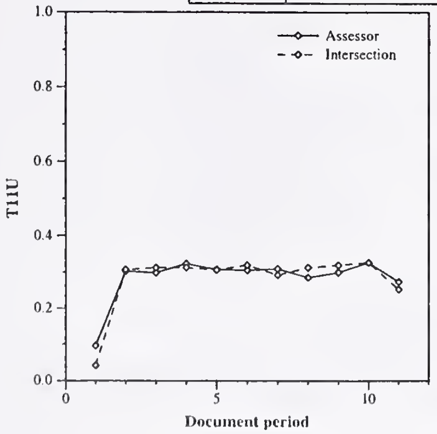
T11U within time periods



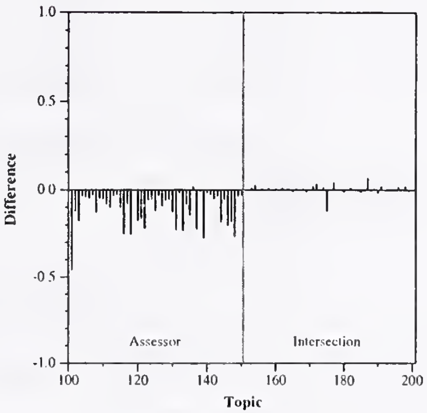
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		dimacs11aP1Q
Subtask		adaptive
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	650	500
Rel-ret:	129	5
Mean T11U	0.272	0.259
Mean T11F	0.036	0.005
Mean Set Recall	0.020	0.002
Mean Set Precision	0.063	0.010
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.096	0.042
2	158952 - 226929	0.303	0.307
3	226930 - 282799	0.299	0.313
4	282800 - 349081	0.324	0.313
5	349082 - 412613	0.308	0.307
6	412614 - 477886	0.306	0.320
7	477887 - 552759	0.310	0.293
8	552760 - 628903	0.286	0.313
9	628904 - 697981	0.300	0.320
10	697982 - 771567	0.327	0.327
11	771568 - 810597	0.274	0.253



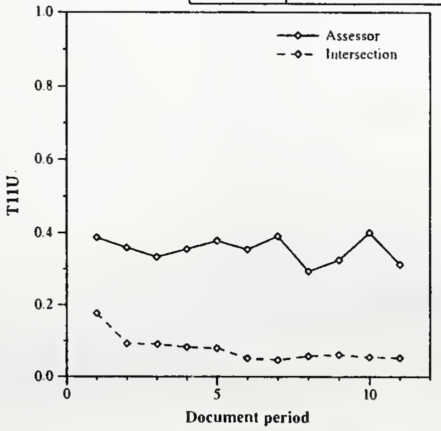
T11U within time periods



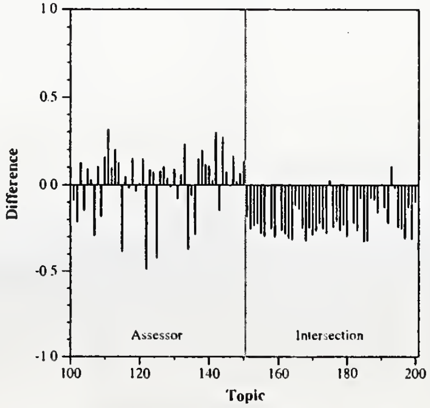
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		thuT11af1
Subtask		adaptive
Resources used:		other Reuters, other data
Optimized for		T11F
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	13202	24611
Rel-ret:	1671	574
Mean T11U	0.395	0.059
Mean T11F	0.417	0.040
Mean Set Recall	0.367	0.101
Mean Set Precision	0.512	0.038
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.387	0.176
2	158952 - 226929	0.359	0.092
3	226930 - 282799	0.333	0.090
4	282800 - 349081	0.355	0.082
5	349082 - 412613	0.378	0.079
6	412614 - 477886	0.354	0.051
7	477887 - 552759	0.391	0.046
8	552760 - 628903	0.294	0.057
9	628904 - 697981	0.325	0.061
10	697982 - 771567	0.401	0.054
11	771568 - 810597	0.313	0.052



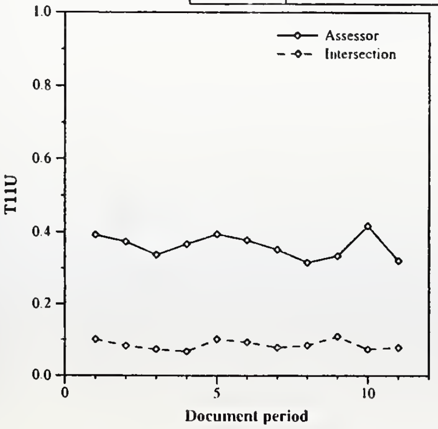
T11U within time periods



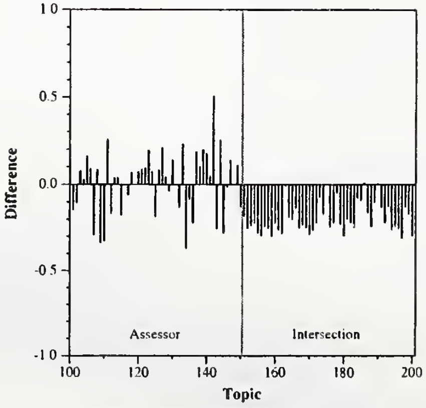
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		thuT11af2
Subtask		adaptive
Resources used:		other Reuters
Optimized for		T11F
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	5814	10491
Rel-ret:	1762	321
Mean T11U	0.389	0.061
Mean T11F	0.422	0.052
Mean Set Recall	0.417	0.065
Mean Set Precision	0.474	0.057
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.393	0.101
2	158952 - 226929	0.374	0.083
3	226930 - 282799	0.337	0.073
4	282800 - 349081	0.367	0.067
5	349082 - 412613	0.395	0.101
6	412614 - 477886	0.378	0.093
7	477887 - 552759	0.352	0.078
8	552760 - 628903	0.316	0.084
9	628904 - 697981	0.334	0.109
10	697982 - 771567	0.418	0.073
11	771568 - 810597	0.321	0.078



T11U within time periods

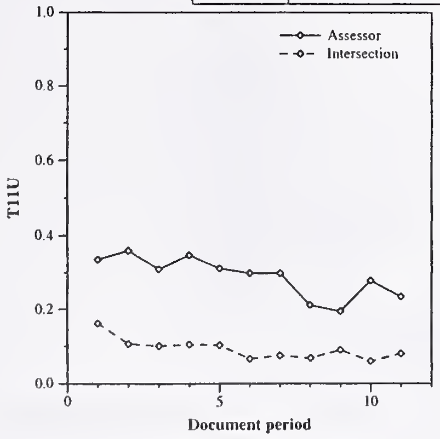


Difference from Median in T11U per Topic

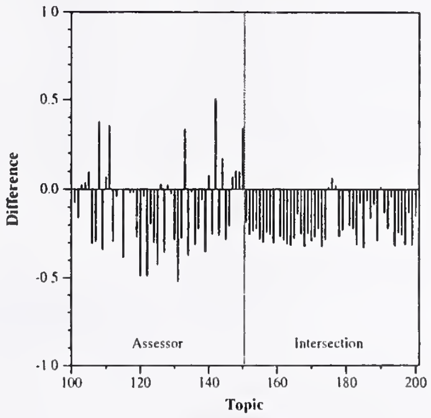
Filtering - Adaptive filtering results — Tsinghua Univ.

Summary Statistics		
Run ID:		thuT11af3
Subtask		adaptive
Resources used:		other Reuters
Optimized for		T11F
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	11042	22060
Rel-ret:	2088	317
Mean T11U	0.277	0.052
Mean T11F	0.337	0.030
Mean Set Recall	0.504	0.060
Mean Set Precision	0.356	0.037
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.336	0.161
2	158952 - 226929	0.360	0.107
3	226930 - 282799	0.310	0.101
4	282800 - 349081	0.348	0.105
5	349082 - 412613	0.312	0.103
6	412614 - 477886	0.299	0.066
7	477887 - 552759	0.299	0.075
8	552760 - 628903	0.212	0.068
9	628904 - 697981	0.195	0.090
10	697982 - 771567	0.279	0.060
11	771568 - 810597	0.235	0.080



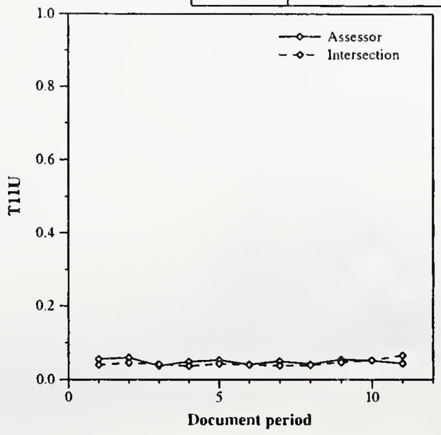
T11U within time periods



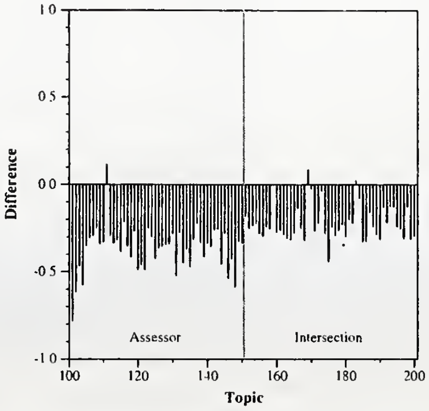
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		cedar02affb0
Subtask		adaptive
Resources used:		none
Optimized for		T11F
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	1973263	2794966
Rel-ret:	1020	562
Mean T11U	0.013	0.034
Mean T11F	0.014	0.003
Mean Set Recall	0.256	0.155
Mean Set Precision	0.021	0.004
Zero returns	1	2

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.056	0.040
2	158952 - 226929	0.060	0.046
3	226930 - 282799	0.038	0.042
4	282800 - 349081	0.049	0.037
5	349082 - 412613	0.053	0.043
6	412614 - 477886	0.041	0.040
7	477887 - 552759	0.050	0.038
8	552760 - 628903	0.042	0.039
9	628904 - 697981	0.055	0.047
10	697982 - 771567	0.052	0.053
11	771568 - 810597	0.044	0.066



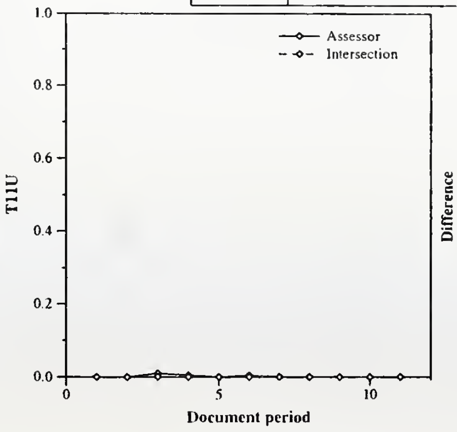
T11U within time periods



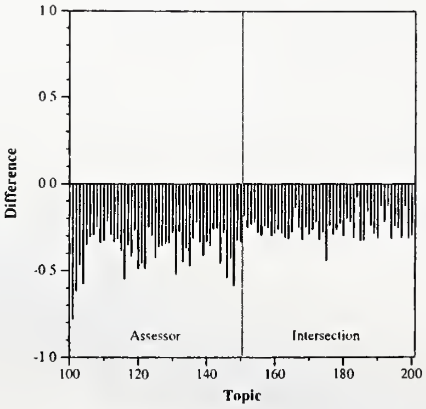
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		cedar02afut0
Subtask		adaptive
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2389916	5139853
Rel-ret:	3368	2409
Mean T11U	0.000	0.000
Mean T11F	0.005	0.001
Mean Set Recall	0.554	0.498
Mean Set Precision	0.004	0.001
Zero returns	0	0

T11U within time periods			
Period	Doc Range	Assessor	Intersection
1	86968 - 158951	0.000	0.000
2	158952 - 226929	0.000	0.000
3	226930 - 282799	0.010	0.001
4	282800 - 349081	0.005	0.000
5	349082 - 412613	0.000	0.000
6	412614 - 477886	0.004	0.000
7	477887 - 552759	0.000	0.000
8	552760 - 628903	0.000	0.000
9	628904 - 697981	0.000	0.000
10	697982 - 771567	0.000	0.000
11	771568 - 810597	0.000	0.000

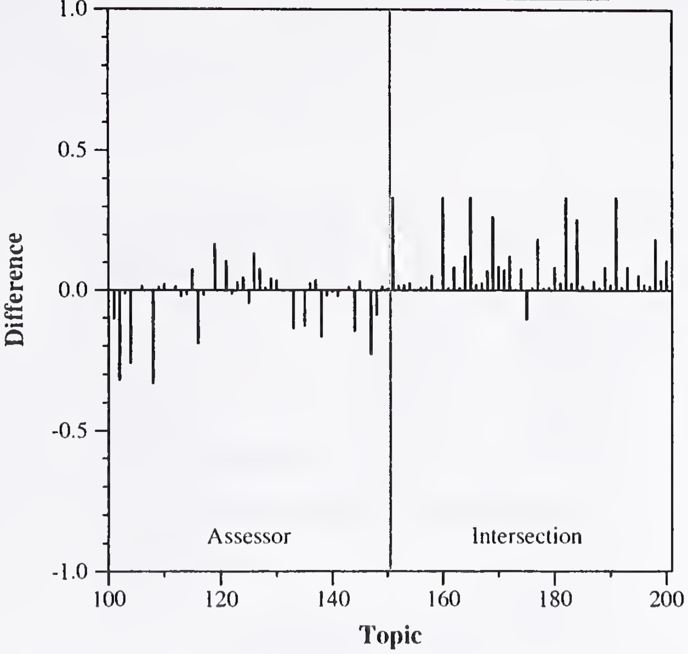


T11U within time periods



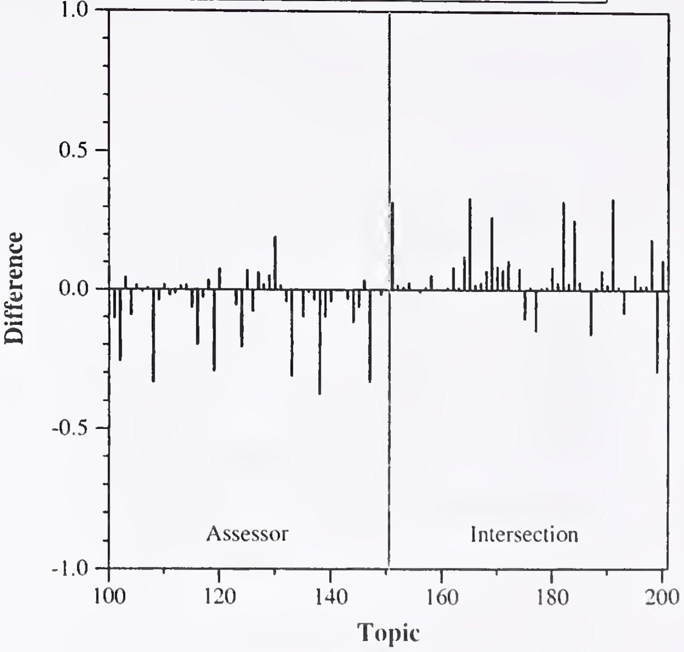
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		ICTBatFT11Ua
Subtask		batch
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	991	
Rel-ret:	540	
Mean T11U	0.350	0.333
Mean T11F	0.180	0.000
Mean Set Recall	0.098	0.000
Mean Set Precision	0.445	0.000
Zero returns	16	50



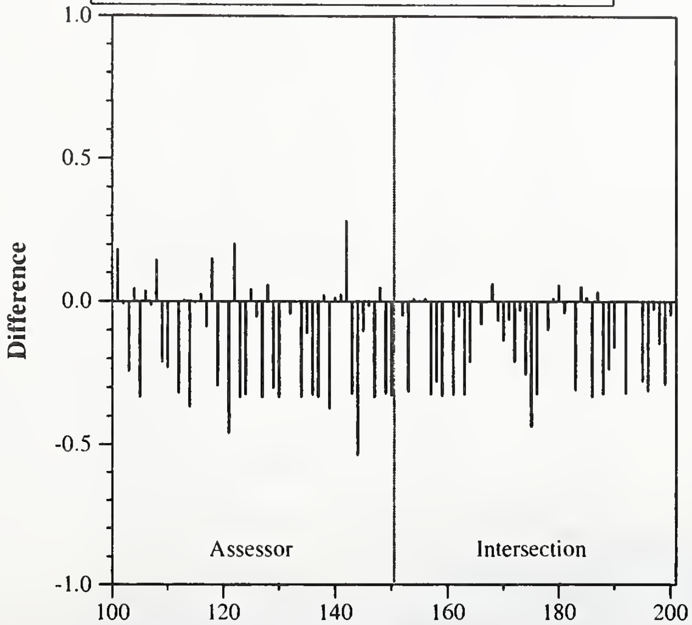
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		ICTBatFT11Ub
Subtask		batch
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	4098	1213
Rel-ret:	1146	66
Mean T11U	0.323	0.305
Mean T11F	0.248	0.011
Mean Set Recall	0.205	0.019
Mean Set Precision	0.432	0.023
Zero returns	1	36



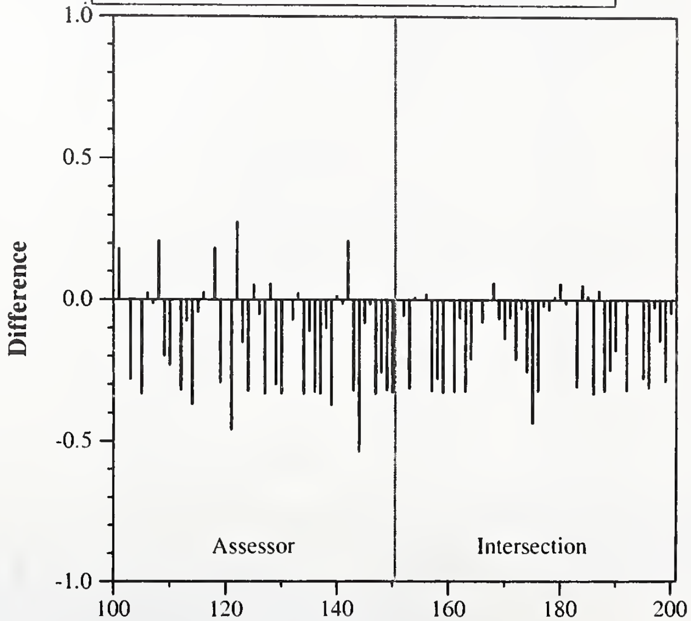
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		CCT11BFC
Subtask		batch
Resources used:		other data
Optimized for		neither
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	40191	32064
Rel-ret:	1931	664
Mean T11U	0.248	0.124
Mean T11F	0.262	0.033
Mean Set Recall	0.448	0.125
Mean Set Precision	0.304	0.042
Zero returns	2	4



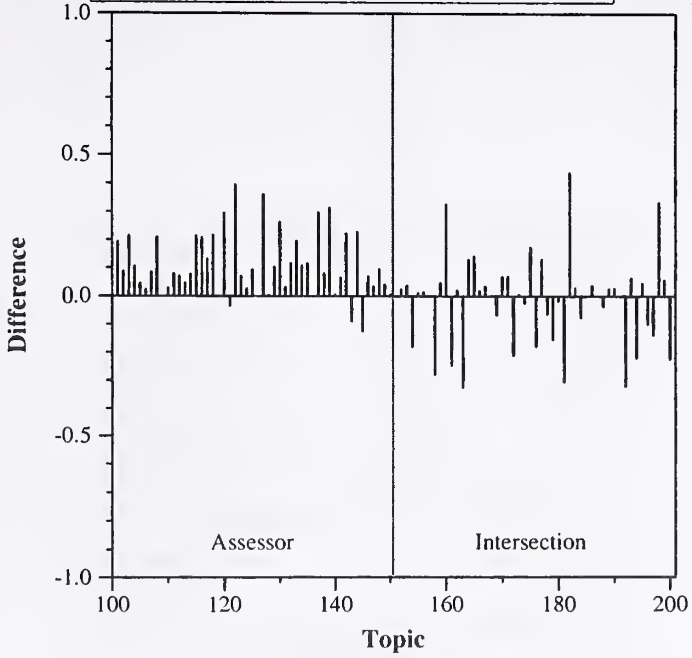
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		CCT11BFD
Subtask		batch
Resources used:		other data
Optimized for		neither
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	40755	33088
Rel-ret:	1970	679
Mean T11U	0.243	0.125
Mean T11F	0.259	0.032
Mean Set Recall	0.453	0.127
Mean Set Precision	0.294	0.041
Zero returns	2	3



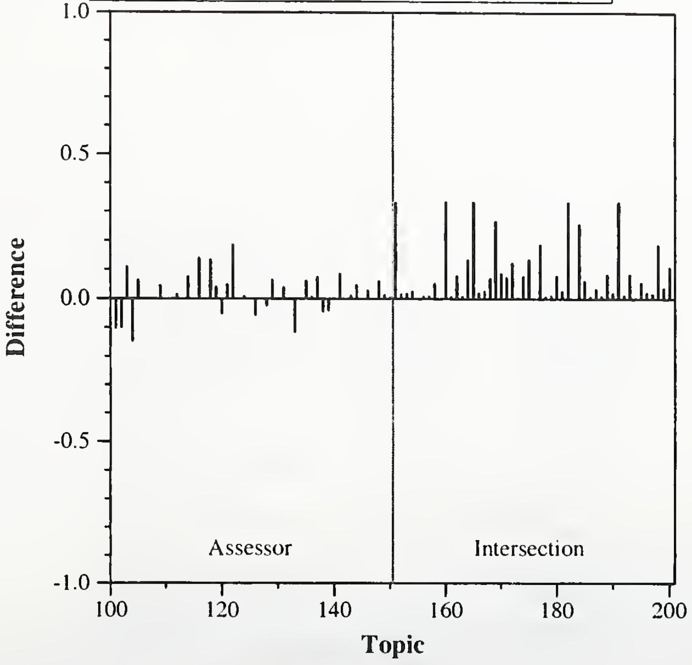
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		iritsigb
Subtask		batch
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	2637	3127
Rel-ret:	1736	280
Mean T11U	0.485	0.237
Mean T11F	0.455	0.091
Mean Set Recall	0.321	0.045
Mean Set Precision	0.661	0.289
Zero returns	0	0



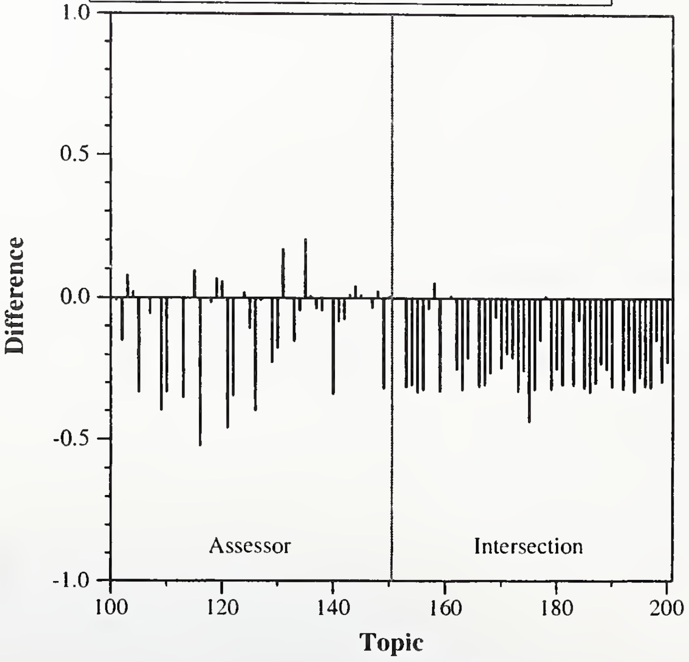
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		apl11FbF
Subtask		batch
Resources used:		other TREC
Optimized for		T11F
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	974	258
Rel-ret:	744	168
Mean T11U	0.391	0.338
Mean T11F	0.216	0.026
Mean Set Recall	0.117	0.013
Mean Set Precision	0.409	0.068
Zero returns	24	41



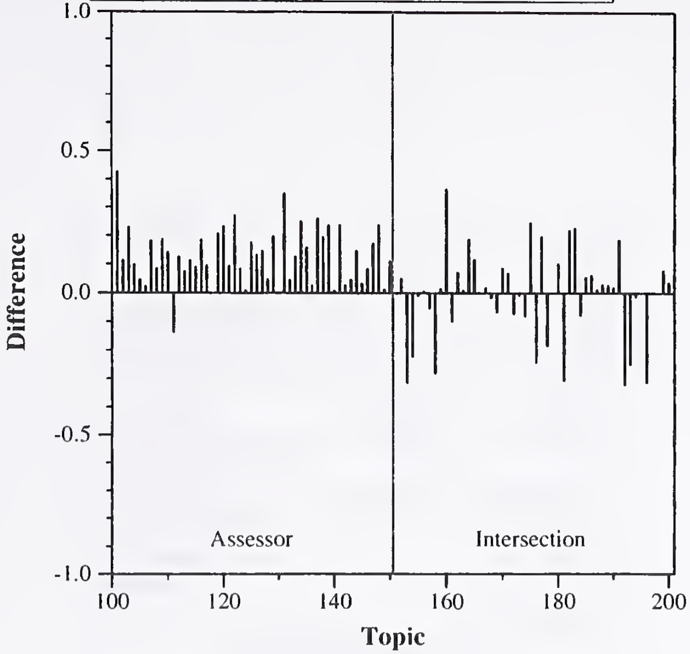
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		apl11FbSU
Subtask		batch
Resources used:		other TREC
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	16102	98670
Rel-ret:	2543	1681
Mean T11U	0.293	0.035
Mean T11F	0.181	0.028
Mean Set Recall	0.255	0.275
Mean Set Precision	0.244	0.027
Zero returns	10	3



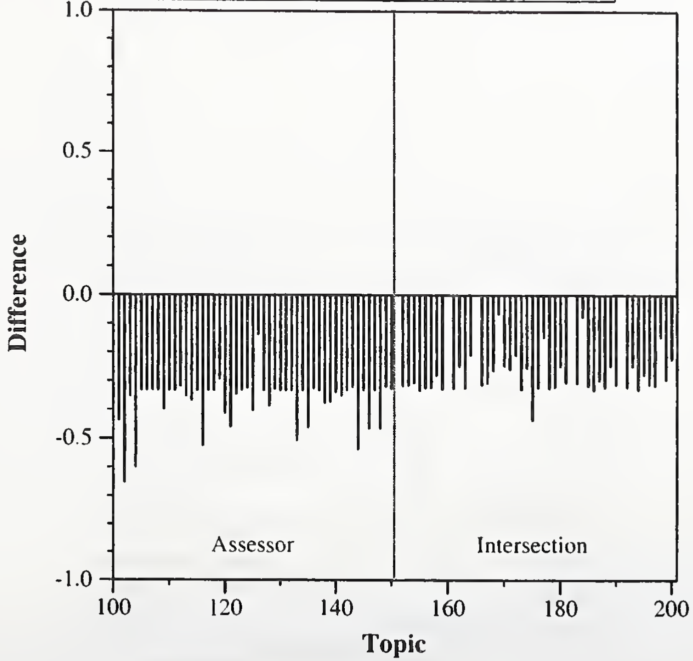
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		KerMITT11bf2
Subtask		batch
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	3037	8572
Rel-ret:	2193	474
Mean T11U	0.505	0.245
Mean T11F	0.495	0.101
Mean Set Recall	0.367	0.060
Mean Set Precision	0.622	0.186
Zero returns	2	3



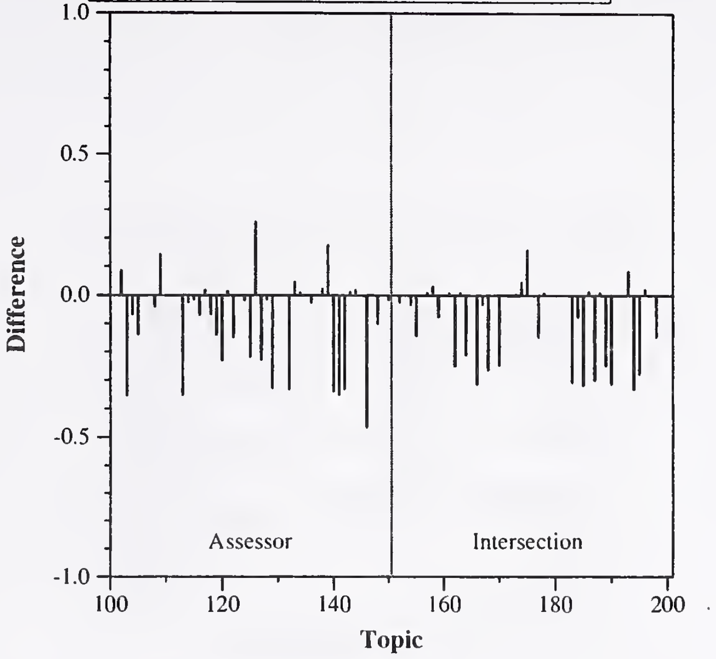
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		mma2002003
Subtask		batch
Resources used:		other data
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	68158	113175
Rel-ret:	3330	1555
Mean T11U	0.005	0.000
Mean T11F	0.056	0.017
Mean Set Recall	0.576	0.349
Mean Set Precision	0.046	0.013
Zero returns	0	0



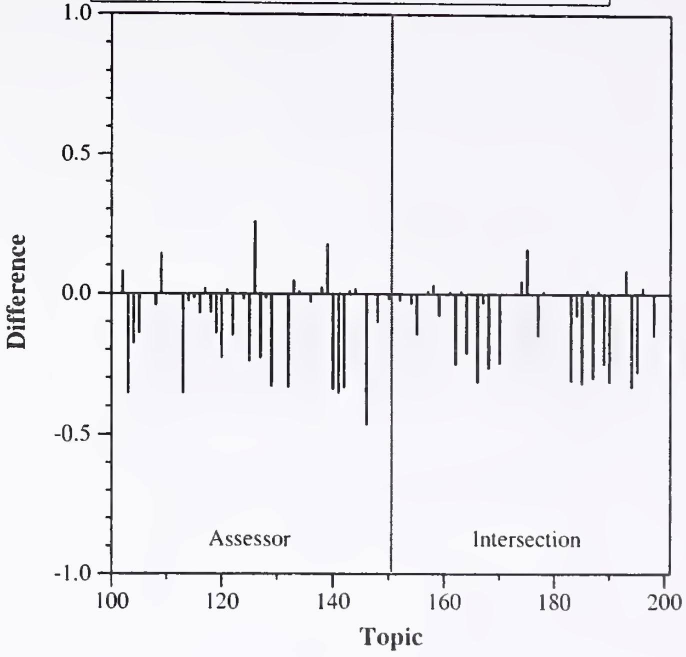
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		kNII11bf1
Subtask		batch
Resources used:		none
Optimized for		neither
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	55658	124758
Rel-ret:	1579	834
Mean T11U	0.305	0.181
Mean T11F	0.190	0.050
Mean Set Recall	0.177	0.146
Mean Set Precision	0.285	0.061
Zero returns	11	8



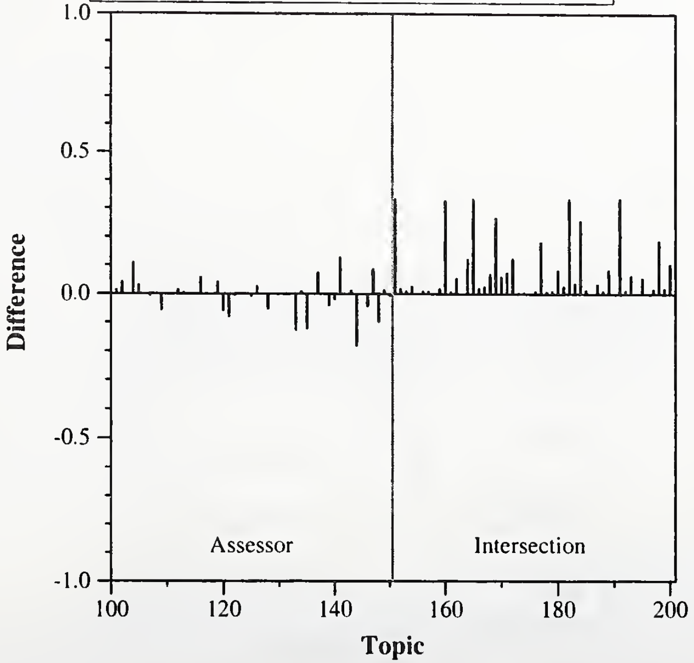
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		kNII11bf2
Subtask		batch
Resources used:		none
Optimized for		neither
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	55716	124784
Rel-ret:	1584	836
Mean T11U	0.302	0.181
Mean T11F	0.188	0.050
Mean Set Recall	0.179	0.147
Mean Set Precision	0.283	0.061
Zero returns	11	8



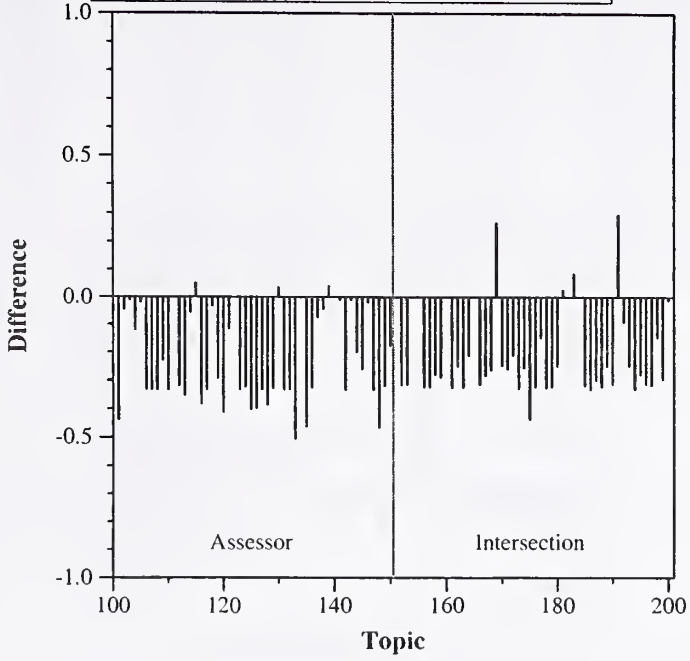
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		dimacs11b001
Subtask		batch
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	698	212
Rel-ret:	535	85
Mean T11U	0.372	0.330
Mean T11F	0.164	0.013
Mean Set Recall	0.073	0.006
Mean Set Precision	0.374	0.051
Zero returns	20	31



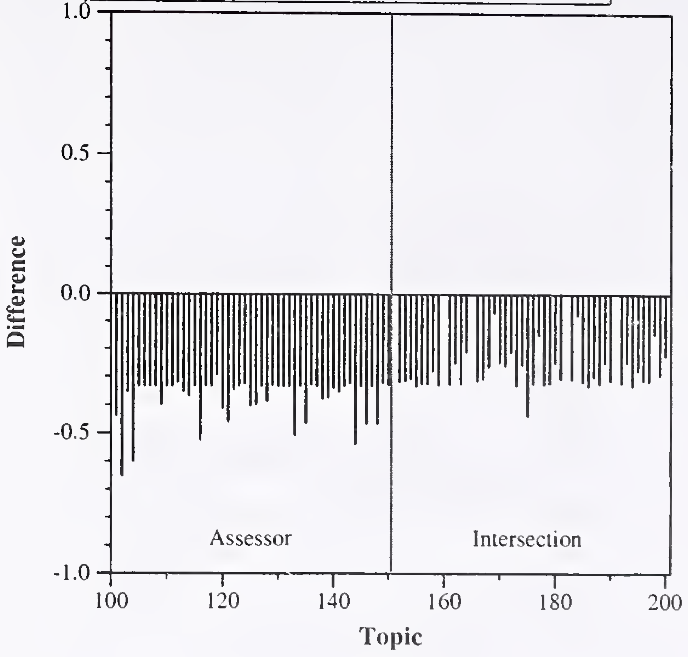
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		cedar02bffb0
Subtask		batch
Resources used:		none
Optimized for		T11F
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	365633	323066
Rel-ret:	1672	1486
Mean T11U	0.155	0.052
Mean T11F	0.090	0.030
Mean Set Recall	0.165	0.211
Mean Set Precision	0.205	0.030
Zero returns	2	3



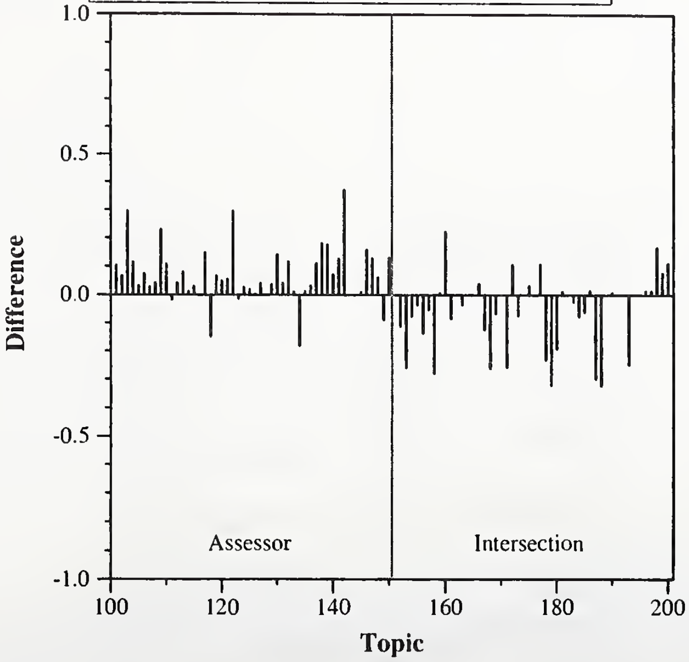
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		cedar02bfut0
Subtask		batch
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	5533383	11820841
Rel-ret:	4636	2968
Mean T11U	0.000	0.000
Mean T11F	0.001	0.001
Mean Set Recall	0.887	0.679
Mean Set Precision	0.001	0.001
Zero returns	0	0



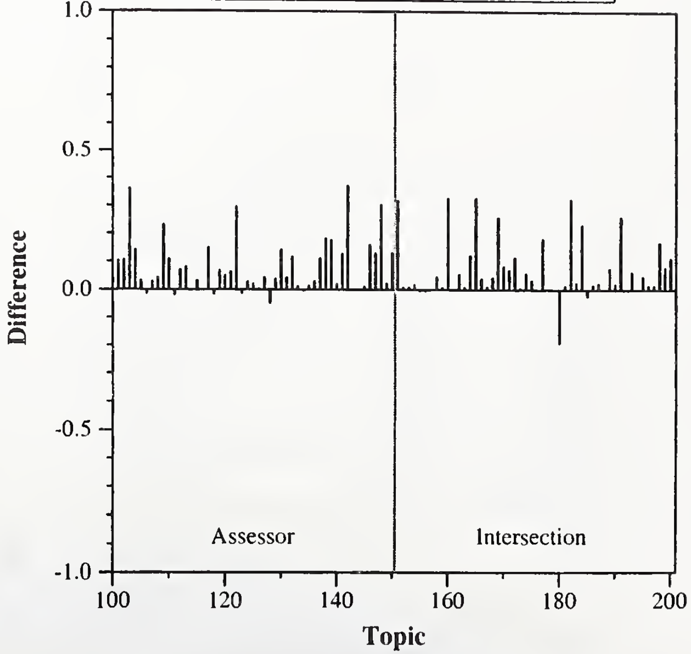
Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		UNTextCatF
Subtask		batch
Resources used:		none
Optimized for		T11F
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	1712	6158
Rel-ret:	1256	322
Mean T11U	0.446	0.200
Mean T11F	0.444	0.083
Mean Set Recall	0.237	0.060
Mean Set Precision	0.682	0.120
Zero returns	0	0



Difference from Median in T11U per Topic

Summary Statistics		
Run ID:		UNTextCatSU
Subtask		batch
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	1713	580
Rel-ret:	1358	194
Mean T11U	0.458	0.322
Mean T11F	0.415	0.052
Mean Set Recall	0.220	0.023
Mean Set Precision	0.693	0.117
Zero returns	0	0



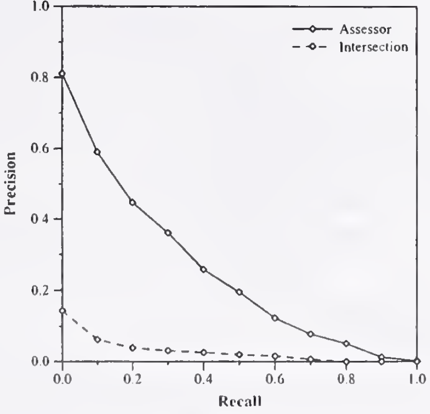
Difference from Median in T11U per Topic

Filtering - Routing filtering results — Chinese Academy of Sciences

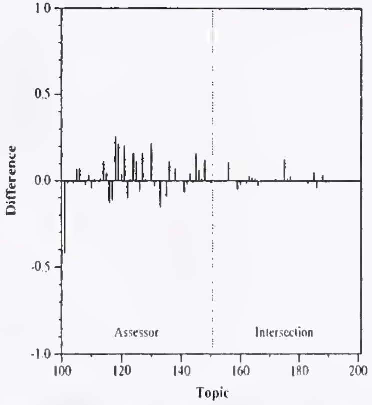
Summary Statistics		
Run ID:		ICTRouFT11Ua
Subtask		routing
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	50000	50000
Rel-ret:	3216	963

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	0.8110	0.1437
0.10	0.5901	0.0624
0.20	0.4473	0.0393
0.30	0.3610	0.0314
0.40	0.2585	0.0262
0.50	0.1955	0.0202
0.60	0.1235	0.0157
0.70	0.0782	0.0071
0.80	0.0513	0.0000
0.90	0.0129	0.0000
1.00	0.0025	0.0000
uninterp. MAP	0.243	0.024

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.5880	0.0680
At 10 docs	0.5220	0.0720
At 15 docs	0.4640	0.0653
At 20 docs	0.4350	0.0660
At 30 docs	0.3967	0.0633
At 100 docs	0.2544	0.0500
At 200 docs	0.1772	0.0422
At 500 docs	0.1048	0.0283
At 1000 docs	0.0643	0.0193
Exact RPrec	0.2940	0.0472



Recall-Precision Curve

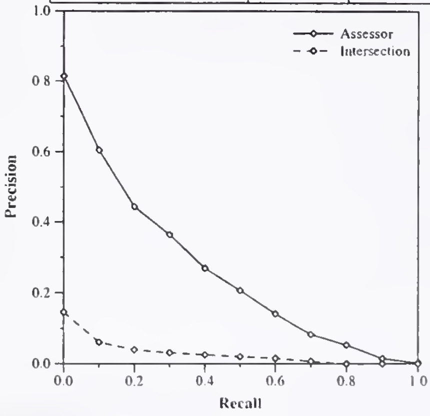


Difference from Median in Average Precision per Topic

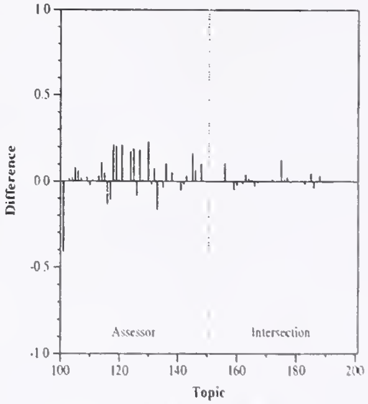
Summary Statistics		
Run ID:		ICTRouFT11Ub
Subtask		routing
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	50000	50000
Rel-ret:	3303	988

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	0.8144	0.1459
0.10	0.6052	0.0616
0.20	0.4438	0.0401
0.30	0.3639	0.0316
0.40	0.2688	0.0260
0.50	0.2068	0.0203
0.60	0.1412	0.0155
0.70	0.0835	0.0071
0.80	0.0536	0.0000
0.90	0.0149	0.0000
1.00	0.0026	0.0000
uninterp. MAP	0.250	0.025

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.5920	0.0760
At 10 docs	0.5260	0.0740
At 15 docs	0.4720	0.0693
At 20 docs	0.4410	0.0690
At 30 docs	0.4027	0.0660
At 100 docs	0.2588	0.0496
At 200 docs	0.1823	0.0439
At 500 docs	0.1064	0.0285
At 1000 docs	0.0661	0.0198
Exact RPrec	0.3020	0.0472



Recall-Precision Curve

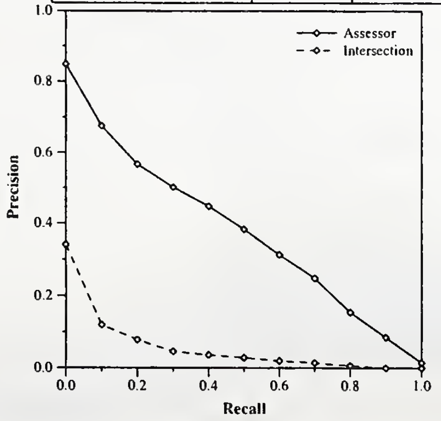


Difference from Median in Average Precision per Topic

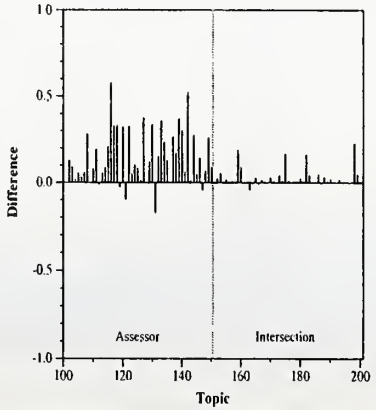
Summary Statistics		
Run ID:		iritsigr
Subtask		routing
Resources used:		other TREC
Optimized for		neither
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	50000	50000
Rel-ret:	4050	1205

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	0.8486	0.3403
0.10	0.6754	0.1194
0.20	0.5665	0.0782
0.30	0.5016	0.0462
0.40	0.4475	0.0362
0.50	0.3831	0.0288
0.60	0.3129	0.0200
0.70	0.2486	0.0146
0.80	0.1549	0.0070
0.90	0.0852	0.0000
1.00	0.0152	0.0000
uninterp. MAP	0.369	0.044

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.6720	0.1640
At 10 docs	0.6260	0.1240
At 15 docs	0.5680	0.1053
At 20 docs	0.5300	0.0950
At 30 docs	0.4813	0.0873
At 100 docs	0.3348	0.0668
At 200 docs	0.2345	0.0558
At 500 docs	0.1374	0.0359
At 1000 docs	0.0810	0.0241
Exact RPrec	0.4122	0.0717



Recall-Precision Curve



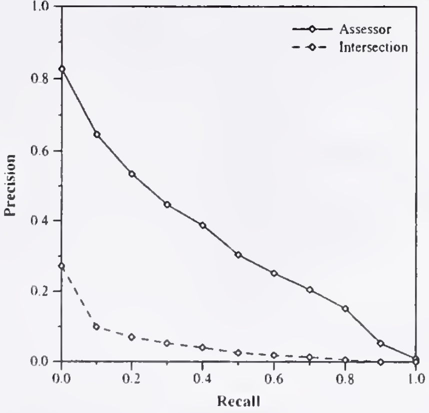
Difference from Median in Average Precision per Topic

Filtering - Routing filtering results — Johns Hopkins University

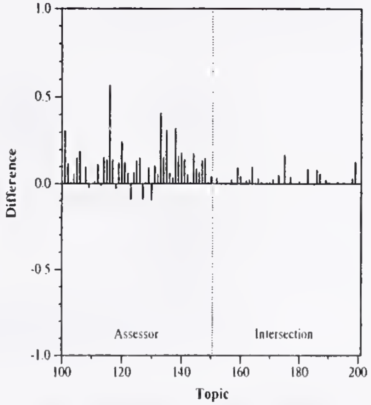
Summary Statistics		
Run ID:		apl11Frm
Subtask		routing
Resources used:		other TREC
Optimized for		neither
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	50000	50000
Rel-ret:	4167	1343

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	0.8271	0.2722
0.10	0.6466	0.0987
0.20	0.5349	0.0699
0.30	0.4468	0.0530
0.40	0.3877	0.0407
0.50	0.3044	0.0258
0.60	0.2524	0.0191
0.70	0.2061	0.0137
0.80	0.1527	0.0062
0.90	0.0531	0.0000
1.00	0.0093	0.0000
uninterp. MAP	0.330	0.042

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.6480	0.1200
At 10 docs	0.5940	0.1020
At 15 docs	0.5440	0.1027
At 20 docs	0.5180	0.1010
At 30 docs	0.4773	0.0947
At 100 docs	0.3180	0.0778
At 200 docs	0.2356	0.0640
At 500 docs	0.1387	0.0421
At 1000 docs	0.0833	0.0269
Exact RPrec	0.3537	0.0765



Recall-Precision Curve

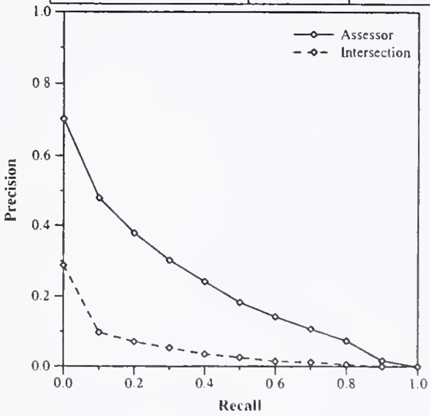


Difference from Median in Average Precision per Topic

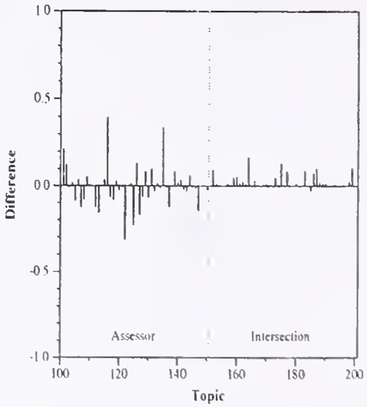
Summary Statistics		
Run ID:		apl11Frsvm
Subtask		routing
Resources used:		other TREC
Optimized for		neither
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	50000	50000
Rel-ret:	3281	1311

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	0.7025	0.2875
0.10	0.4793	0.0973
0.20	0.3783	0.0703
0.30	0.3021	0.0530
0.40	0.2419	0.0352
0.50	0.1824	0.0258
0.60	0.1425	0.0149
0.70	0.1071	0.0122
0.80	0.0733	0.0058
0.90	0.0168	0.0000
1.00	0.0000	0.0000
uninterp. MAP	0.218	0.043

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.5320	0.1440
At 10 docs	0.4620	0.1200
At 15 docs	0.4267	0.1187
At 20 docs	0.4130	0.1200
At 30 docs	0.3740	0.1087
At 100 docs	0.2476	0.0790
At 200 docs	0.1779	0.0611
At 500 docs	0.1051	0.0396
At 1000 docs	0.0656	0.0262
Exact RPrec	0.2556	0.0770



Recall-Precision Curve



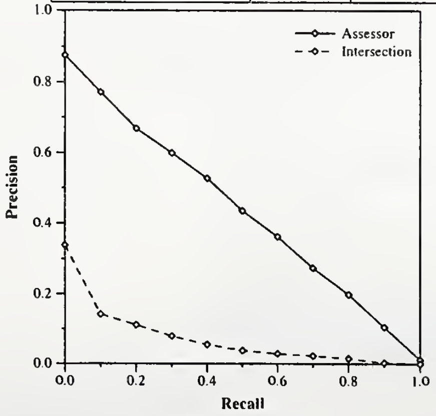
Difference from Median in Average Precision per Topic

Filtering - Routing filtering results — KerMIT Consortium

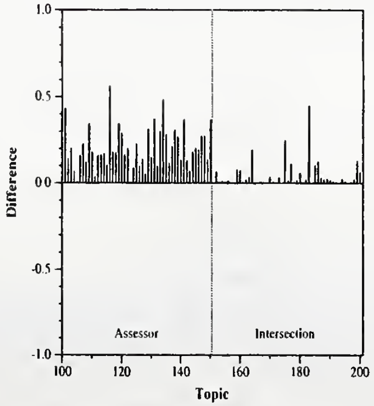
Summary Statistics		
Run ID:		KerMITT11rr2
Subtask		routing
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	50000	50000
Rel-ret:	4263	1540

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	0.8757	0.3381
0.10	0.7712	0.1419
0.20	0.6683	0.1115
0.30	0.5988	0.0790
0.40	0.5271	0.0547
0.50	0.4353	0.0376
0.60	0.3625	0.0288
0.70	0.2743	0.0227
0.80	0.1985	0.0152
0.90	0.1054	0.0031
1.00	0.0115	0.0000
uninterp. MAP	0.427	0.061

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.7240	0.1520
At 10 docs	0.6940	0.1460
At 15 docs	0.6573	0.1507
At 20 docs	0.6320	0.1500
At 30 docs	0.5760	0.1433
At 100 docs	0.3946	0.1088
At 200 docs	0.2741	0.0847
At 500 docs	0.1496	0.0496
At 1000 docs	0.0853	0.0308
Exact RPrec	0.4493	0.1097



Recall-Precision Curve



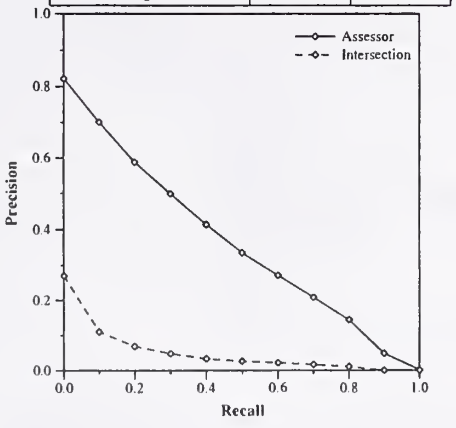
Difference from Median in Average Precision per Topic

Filtering - Routing filtering results — Microsoft Research Cambridge

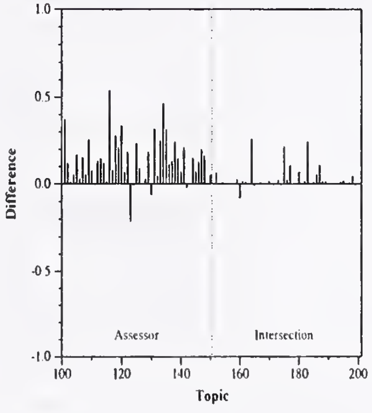
Summary Statistics		
Run ID:		msPUMb
Subtask		routing
Resources used:		other TREC
Optimized for		neither
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	50000	50000
Rel-ret:	3961	1214

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	0.8226	0.2706
0.10	0.7008	0.1102
0.20	0.5883	0.0691
0.30	0.4987	0.0485
0.40	0.4139	0.0336
0.50	0.3360	0.0265
0.60	0.2720	0.0222
0.70	0.2087	0.0168
0.80	0.1438	0.0110
0.90	0.0482	0.0000
1.00	0.0015	0.0000
uninterp. MAP	0.355	0.045

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.6880	0.1440
At 10 docs	0.6400	0.1220
At 15 docs	0.5973	0.1227
At 20 docs	0.5610	0.1230
At 30 docs	0.5100	0.1047
At 100 docs	0.3360	0.0788
At 200 docs	0.2405	0.0610
At 500 docs	0.1363	0.0386
At 1000 docs	0.0792	0.0243
Exact RPrec	0.3748	0.0804



Recall-Precision Curve

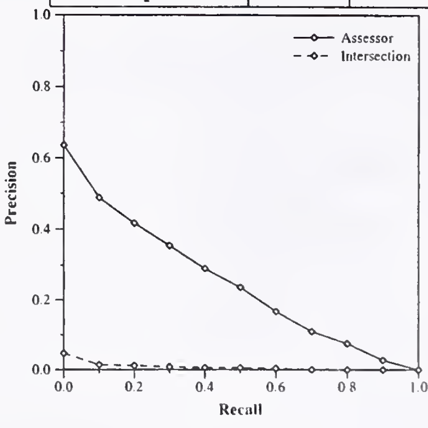


Difference from Median in Average Precision per Topic

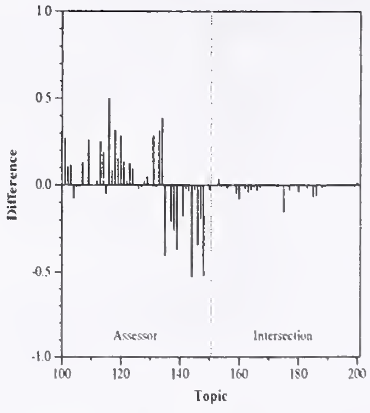
Summary Statistics		
Run ID:		msPUMs
Subtask		routing
Resources used:		other TREC
Optimized for		neither
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	50000	50000
Rel-ret:	2495	266

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	0.6358	0.0470
0.10	0.4878	0.0145
0.20	0.4166	0.0119
0.30	0.3545	0.0079
0.40	0.2905	0.0060
0.50	0.2367	0.0056
0.60	0.1669	0.0039
0.70	0.1092	0.0000
0.80	0.0754	0.0000
0.90	0.0278	0.0000
1.00	0.0000	0.0000
uninterp. MAP	0.239	0.006

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.4880	0.0200
At 10 docs	0.4380	0.0180
At 15 docs	0.4213	0.0200
At 20 docs	0.3980	0.0200
At 30 docs	0.3600	0.0180
At 100 docs	0.2376	0.0134
At 200 docs	0.1662	0.0097
At 500 docs	0.0884	0.0076
At 1000 docs	0.0499	0.0053
Exact RPrec	0.2639	0.0113



Recall-Precision Curve

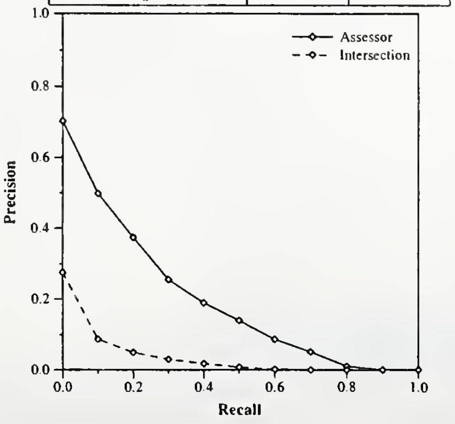


Difference from Median in Average Precision per Topic

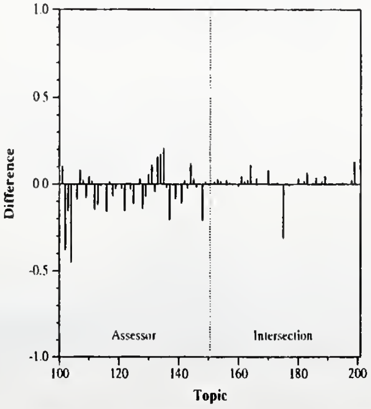
Summary Statistics		
Run ID:		mma2002001
Subtask		routing
Resources used:		other data
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	42906	49790
Rel-ret:	2765	1077

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	0.7036	0.2754
0.10	0.4985	0.0874
0.20	0.3746	0.0498
0.30	0.2558	0.0301
0.40	0.1903	0.0176
0.50	0.1408	0.0077
0.60	0.0872	0.0014
0.70	0.0517	0.0000
0.80	0.0105	0.0000
0.90	0.0000	0.0000
1.00	0.0000	0.0000
uninterp. MAP	0.188	0.029

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.5160	0.1240
At 10 docs	0.4540	0.1060
At 15 docs	0.4293	0.1000
At 20 docs	0.3920	0.0950
At 30 docs	0.3493	0.0933
At 100 docs	0.2314	0.0692
At 200 docs	0.1632	0.0495
At 500 docs	0.0949	0.0324
At 1000 docs	0.0553	0.0215
Exact RPrec	0.2593	0.0610



Recall-Precision Curve

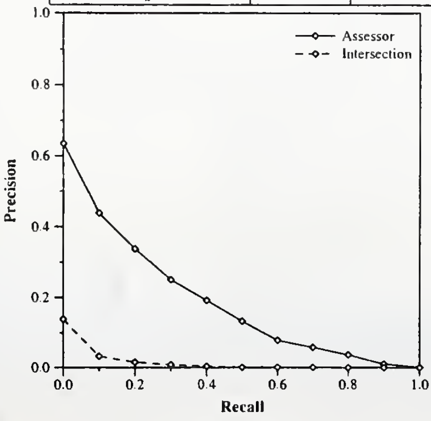


Difference from Median in Average Precision per Topic

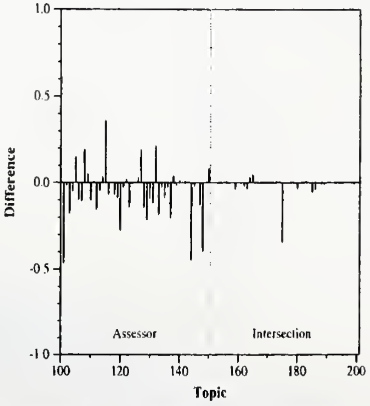
Summary Statistics		
Run ID:		mma2002002
Subtask		routing
Resources used:		other data
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	47985	48059
Rel-ret:	2789	669

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	0.6354	0.1385
0.10	0.4385	0.0333
0.20	0.3379	0.0157
0.30	0.2507	0.0087
0.40	0.1928	0.0041
0.50	0.1335	0.0012
0.60	0.0791	0.0012
0.70	0.0590	0.0008
0.80	0.0375	0.0000
0.90	0.0104	0.0000
1.00	0.0010	0.0000
uninterp. MAP	0.171	0.010

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.4320	0.0600
At 10 docs	0.4160	0.0540
At 15 docs	0.3947	0.0440
At 20 docs	0.3540	0.0420
At 30 docs	0.3200	0.0407
At 100 docs	0.1968	0.0340
At 200 docs	0.1451	0.0274
At 500 docs	0.0878	0.0183
At 1000 docs	0.0558	0.0134
Exact RPrec	0.2129	0.0288



Recall-Precision Curve



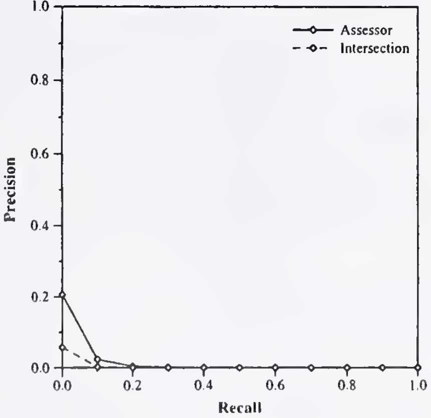
Difference from Median in Average Precision per Topic

Filtering - Routing filtering results — Queensland University of Technology

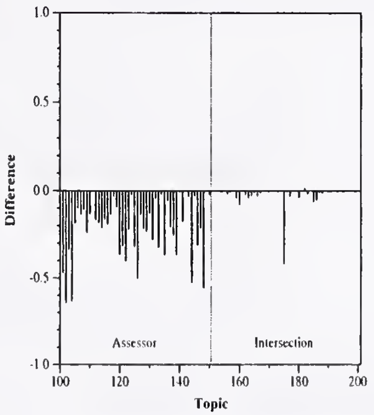
Summary Statistics		
Run ID:		QUT
Subtask		routing
Resources used:		none
Optimized for		T11U
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	29095	40948
Rel-ret:	366	204

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	0.2068	0.0577
0.10	0.0234	0.0023
0.20	0.0043	0.0000
0.30	0.0012	0.0000
0.40	0.0000	0.0000
0.50	0.0000	0.0000
0.60	0.0000	0.0000
0.70	0.0000	0.0000
0.80	0.0000	0.0000
0.90	0.0000	0.0000
1.00	0.0000	0.0000
uninterp. MAP	0.008	0.001

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.0711	0.0160
At 10 docs	0.0800	0.0160
At 15 docs	0.0741	0.0133
At 20 docs	0.0722	0.0120
At 30 docs	0.0630	0.0113
At 100 docs	0.0362	0.0076
At 200 docs	0.0238	0.0062
At 500 docs	0.0130	0.0054
At 1000 docs	0.0081	0.0041
Exact RPrec	0.0303	0.0084



Recall-Precision Curve

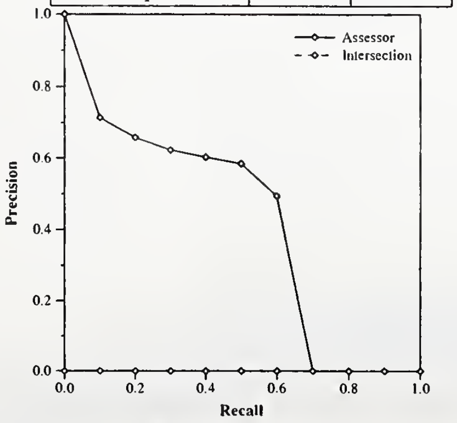


Difference from Median in Average Precision per Topic

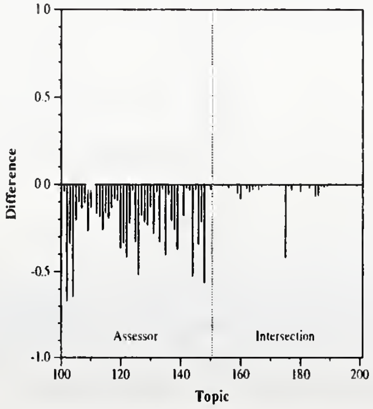
Summary Statistics		
Run ID:		UHcl1
Subtask		routing
Resources used:		other data
Optimized for		neither
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	478	
Rel-ret:	214	

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	1.0000	0.0000
0.10	0.7143	0.0000
0.20	0.6581	0.0000
0.30	0.6229	0.0000
0.40	0.6027	0.0000
0.50	0.5842	0.0000
0.60	0.4935	0.0000
0.70	0.0000	0.0000
0.80	0.0000	0.0000
0.90	0.0000	0.0000
1.00	0.0000	0.0000
uninterp. MAP	0.009	0.000

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.8000	0.0000
At 10 docs	0.8000	0.0000
At 15 docs	0.8000	0.0000
At 20 docs	0.7500	0.0000
At 30 docs	0.6667	0.0000
At 100 docs	0.6400	0.0000
At 200 docs	0.5700	0.0000
At 500 docs	0.4280	0.0000
At 1000 docs	0.2140	0.0000
Exact RPrec	0.5283	0.0000



Recall-Precision Curve

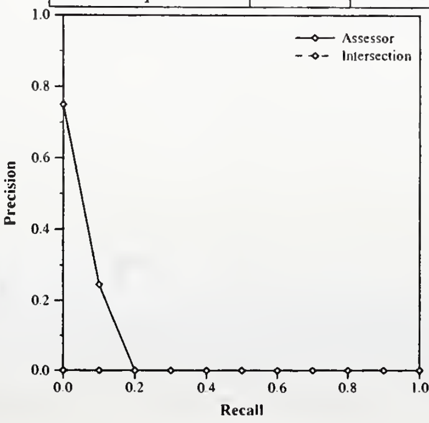


Difference from Median in Average Precision per Topic

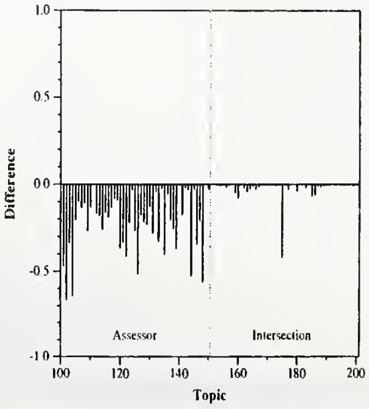
Summary Statistics		
Run ID:		UHcl2
Subtask		routing
Resources used:		other data
Optimized for		neither
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	261	
Rel-ret:	18	

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	0.7500	0.0000
0.10	0.2456	0.0000
0.20	0.0000	0.0000
0.30	0.0000	0.0000
0.40	0.0000	0.0000
0.50	0.0000	0.0000
0.60	0.0000	0.0000
0.70	0.0000	0.0000
0.80	0.0000	0.0000
0.90	0.0000	0.0000
1.00	0.0000	0.0000
uninterp. MAP	0.001	0.000

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.6000	0.0000
At 10 docs	0.6000	0.0000
At 15 docs	0.6667	0.0000
At 20 docs	0.5000	0.0000
At 30 docs	0.4000	0.0000
At 100 docs	0.1500	0.0000
At 200 docs	0.0750	0.0000
At 500 docs	0.0360	0.0000
At 1000 docs	0.0180	0.0000
Exact RPrec	0.1136	0.0000



Recall-Precision Curve

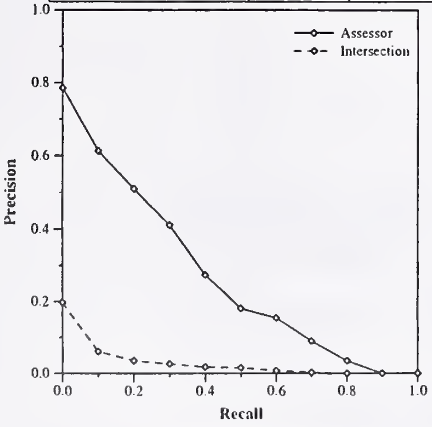


Difference from Median in Average Precision per Topic

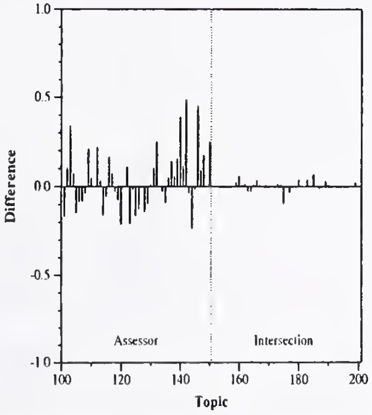
Summary Statistics		
Run ID:		UNTextCatR
Subtask		routing
Resources used:		none
Optimized for		neither
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	25101	38664
Rel-ret:	2541	725

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	0.7861	0.1979
0.10	0.6127	0.0607
0.20	0.5099	0.0352
0.30	0.4112	0.0264
0.40	0.2740	0.0178
0.50	0.1818	0.0155
0.60	0.1553	0.0076
0.70	0.0898	0.0026
0.80	0.0350	0.0000
0.90	0.0000	0.0000
1.00	0.0000	0.0000
uninterp. MAP	0.260	0.023

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.5800	0.0720
At 10 docs	0.5640	0.0800
At 15 docs	0.5200	0.0787
At 20 docs	0.5190	0.0830
At 30 docs	0.4720	0.0733
At 100 docs	0.2708	0.0586
At 200 docs	0.1840	0.0429
At 500 docs	0.0934	0.0245
At 1000 docs	0.0508	0.0145
Exact RPrec	0.3078	0.0531



Recall-Precision Curve

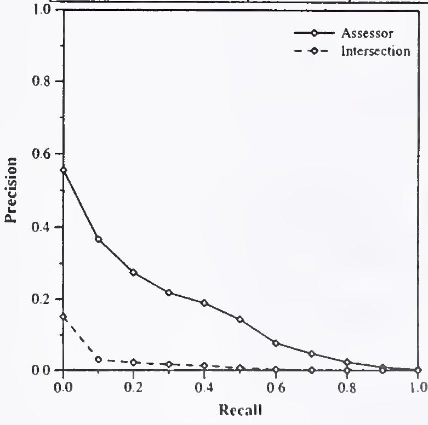


Difference from Median in Average Precision per Topic

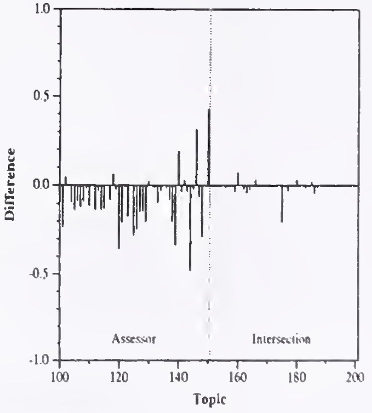
Summary Statistics		
Run ID:		UNTextCatR1
Subtask		routing
Resources used:		none
Optimized for		neither
Topic type:	Assessor	Intersection
Number of Topics:	50	50
Retrieved:	50000	50000
Rel-ret:	2593	662

Recall Level Averages		
Recall	Precision	
	Assr	Intsc
0.00	0.5555	0.1520
0.10	0.3672	0.0299
0.20	0.2754	0.0219
0.30	0.2185	0.0167
0.40	0.1904	0.0134
0.50	0.1448	0.0067
0.60	0.0775	0.0024
0.70	0.0473	0.0010
0.80	0.0235	0.0000
0.90	0.0089	0.0000
1.00	0.0019	0.0000
uninterp. MAP	0.149	0.014

Document Level Averages		
	Precision	
	Assr	Intsc
At 5 docs	0.3760	0.0600
At 10 docs	0.3460	0.0480
At 15 docs	0.3227	0.0427
At 20 docs	0.3050	0.0380
At 30 docs	0.2880	0.0300
At 100 docs	0.1992	0.0300
At 200 docs	0.1406	0.0255
At 500 docs	0.0818	0.0198
At 1000 docs	0.0519	0.0132
Exact RPrec	0.1953	0.0289



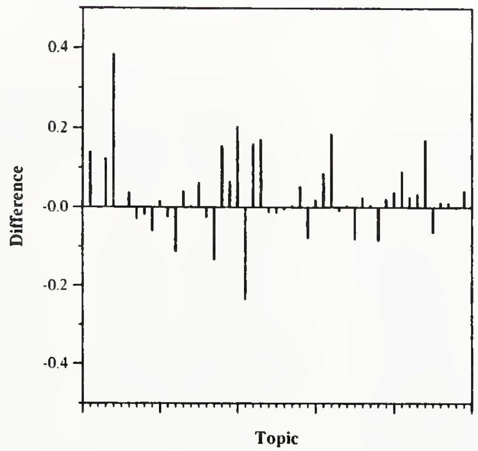
Recall-Precision Curve



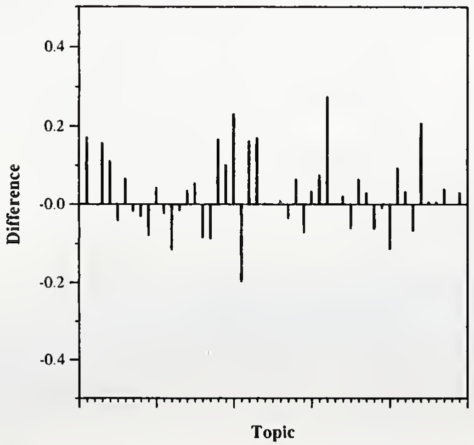
Difference from Median in Average Precision per Topic

Summary Statistics	
Run ID	cmu02t300rAs
Num topics	

	Relevant	New
Average precision	0.17	0.16
Average recall	0.23	0.18
Average F	0.182	0.157



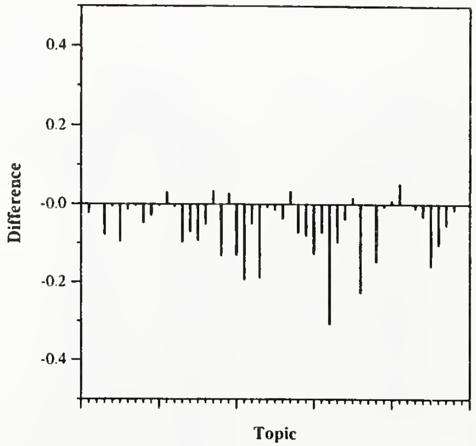
Difference from Median in F score per topic for relevant sentences



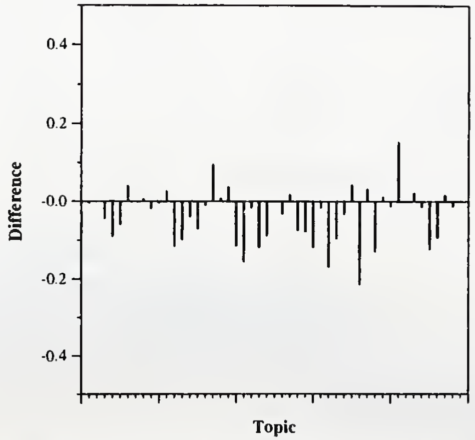
Difference from Median in F score per topic for new sentences

Summary Statistics	
Run ID	cmu02t300rBw
Num topics	

	Relevant	New
Average precision	0.10	0.10
Average recall	0.13	0.13
Average F	0.099	0.094

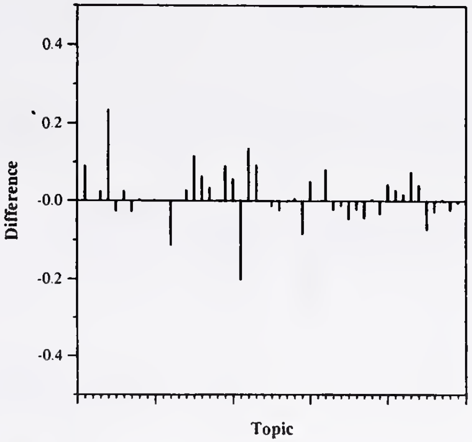


Difference from Median in F score per topic for relevant sentences

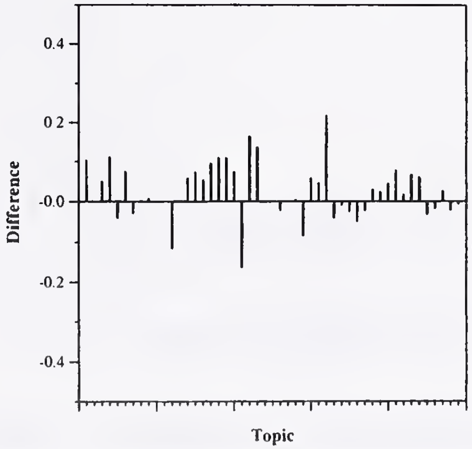


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID	cmu02t300rCb	
Num topics		
	Relevant	New
Average precision	0.13	0.12
Average recall	0.31	0.31
Average F	0.165	0.154

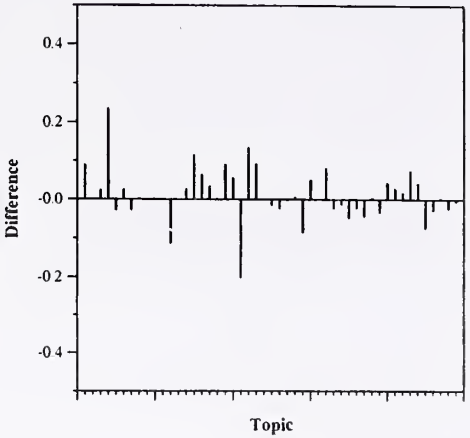


Difference from Median in F score per topic for relevant sentences

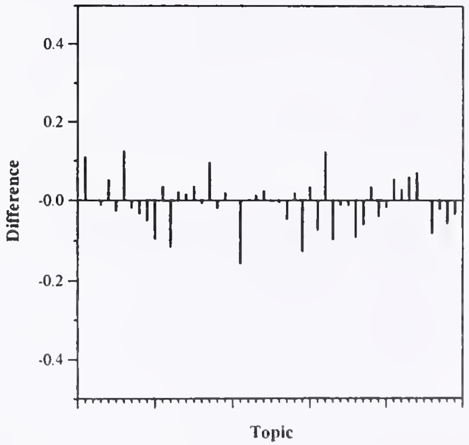


Diffcrence from Median in F score per topic for new sentences

Summary Statistics		
Run ID	cmu02t300rCv	
Num topics		
	Relevant	New
Average precision	0.13	0.12
Average recall	0.31	0.16
Average F	0.165	0.122

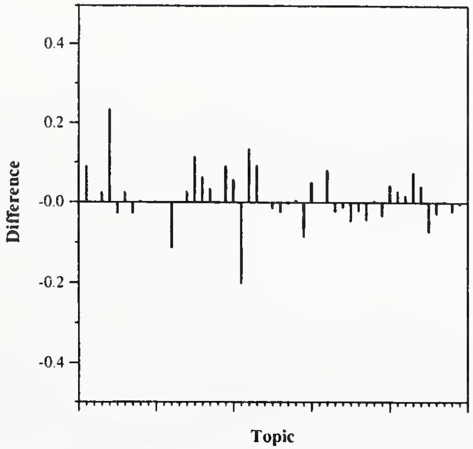


Difference from Median in F score per topic for relevant sentences

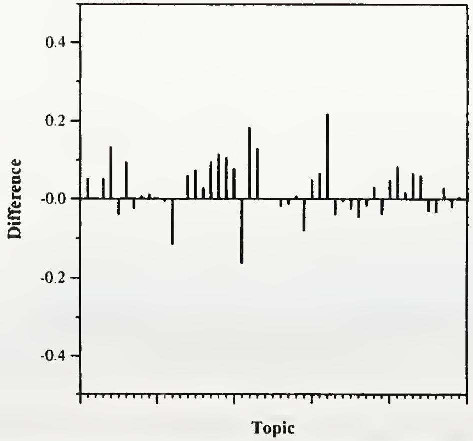


Differencc from Mcdian in F score per topic for new scntences

Summary Statistics		
Run ID	cmu02t300rCw	
Num topics		
	Relevant	New
Average precision	0.13	0.12
Average recall	0.31	0.30
Average F	0.165	0.153

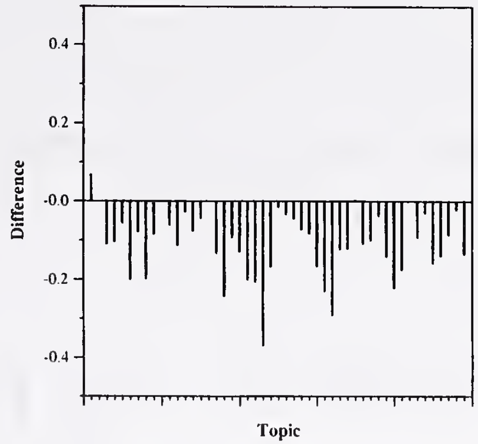


Difference from Median in F score per topic for relevant sentences

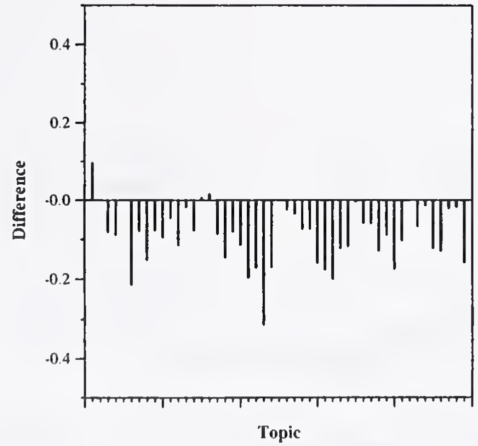


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		novcolcl35
Num topics		
	Relevant	New
Average precision	0.07	0.07
Average recall	0.04	0.04
Average F	0.047	0.040

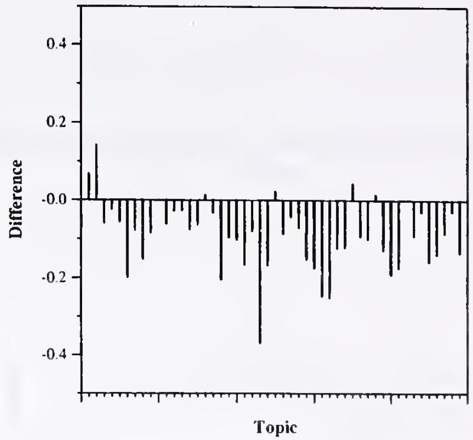


Difference from Median in F score per topic for relevant sentences

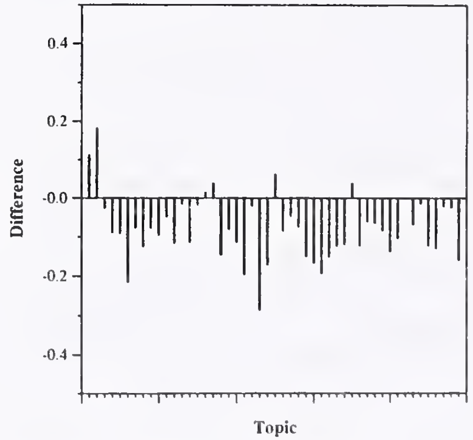


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		novcolcl85
Num topics		
	Relevant	New
Average precision	0.09	0.08
Average recall	0.07	0.05
Average F	0.063	0.050

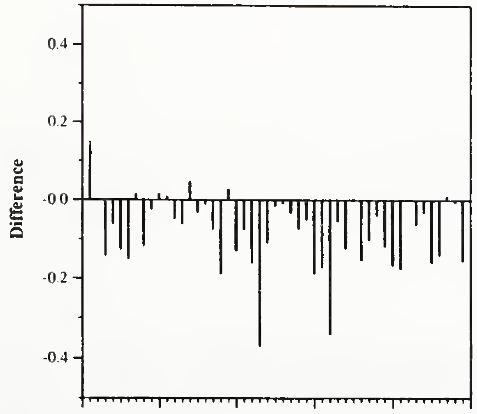


Difference from Median in F score per topic for relevant sentences

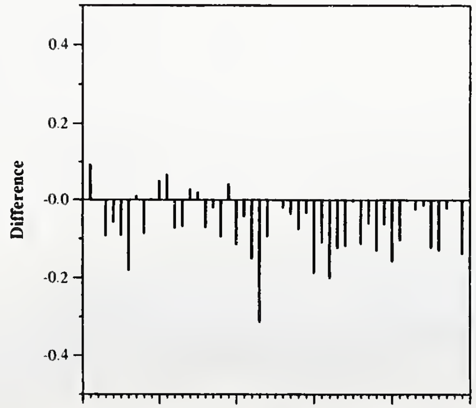


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		novcolclfx
Num topics		
	Relevant	New
Average precision	0.07	0.07
Average recall	0.12	0.09
Average F	0.073	0.063

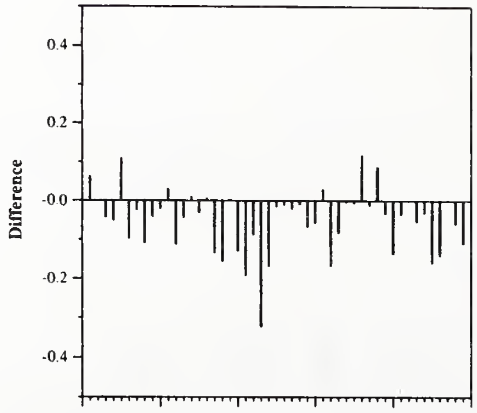


Difference from Median in F score per topic for relevant sentences

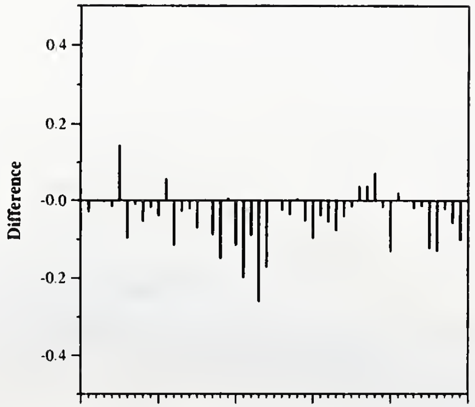


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		novcolmerg
Num topics		
	Relevant	New
Average precision	0.11	0.09
Average recall	0.15	0.10
Average F	0.103	0.083



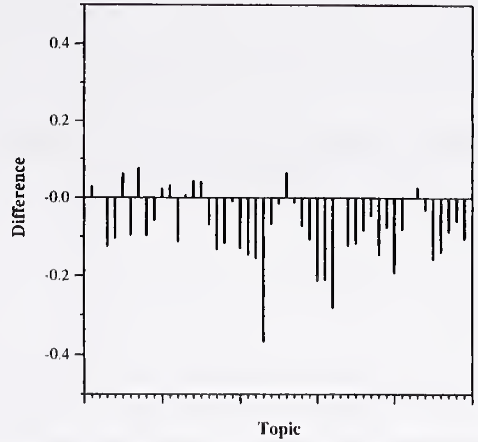
Difference from Median in F score per topic for relevant sentences



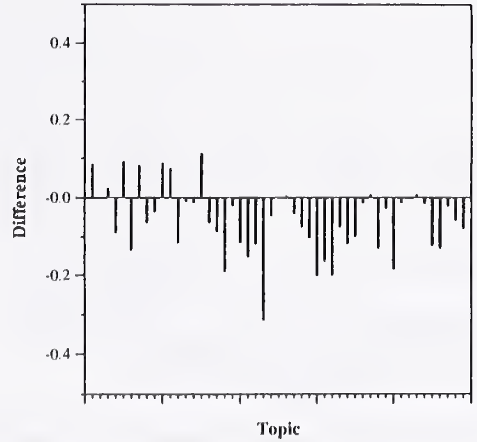
Difference from Median in F score per topic for new sentences

Summary Statistics	
Run ID	novcolsent
Num topics	

	Relevant	New
Average precision	0.11	0.12
Average recall	0.09	0.07
Average F	0.078	0.071

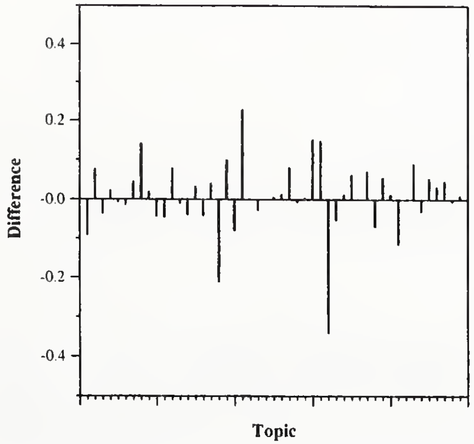


Difference from Median in F score per topic for relevant sentenees

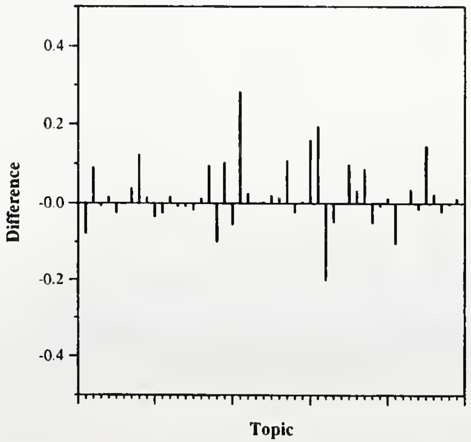


Differenee from Median in F seore per topic for new sentences

Summary Statistics		
Run ID		fdut11n1
Num topics		
	Relevant	New
Average precision	0.14	0.12
Average recall	0.30	0.26
Average F	0.161	0.148

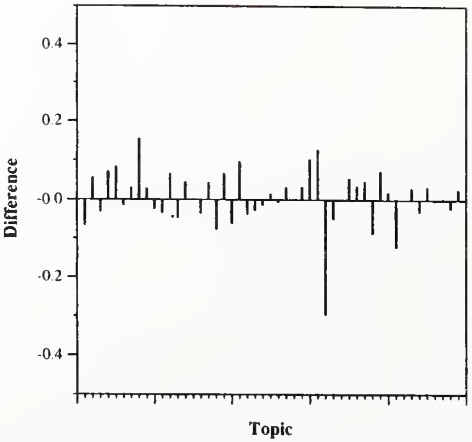


Difference from Median in F score per topic for relevant sentences

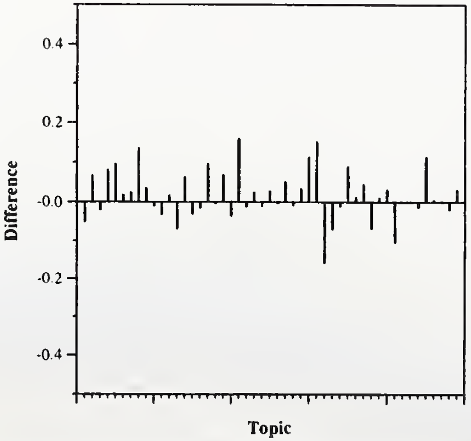


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		fdut11n2
Num topics		
	Relevant	New
Average precision	0.12	0.11
Average recall	0.39	0.34
Average F	0.160	0.147

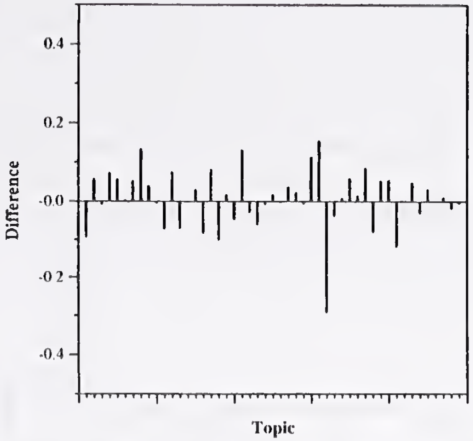


Difference from Median in F score per topic for relevant sentences

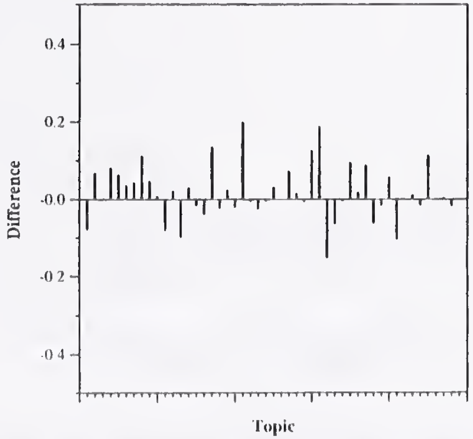


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		fdut11n3
Num topics		
	Relevant	New
Average precision	0.12	0.11
Average recall	0.35	0.31
Average F	0.160	0.146

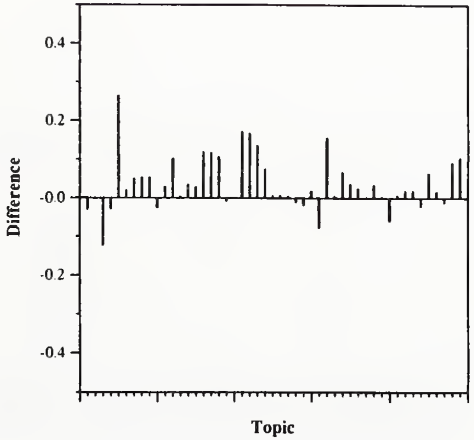


Difference from Median in F score per topic for relevant sentences

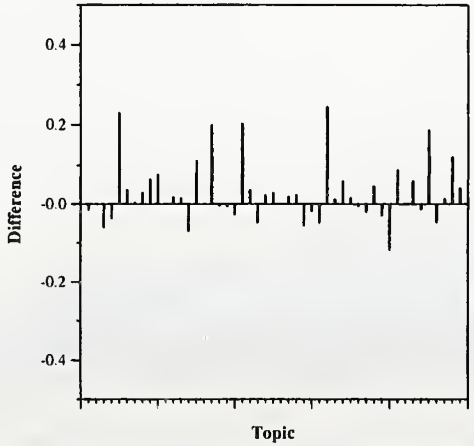


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		dumbrun
Num topics		
	Relevant	New
Average precision	0.15	0.16
Average recall	0.49	0.22
Average F	0.190	0.157

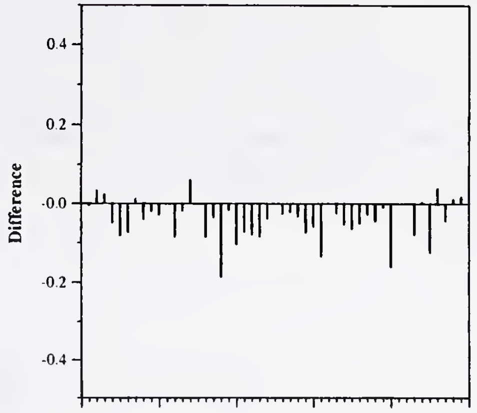


Difference from Median in F score per topic for relevant sentences

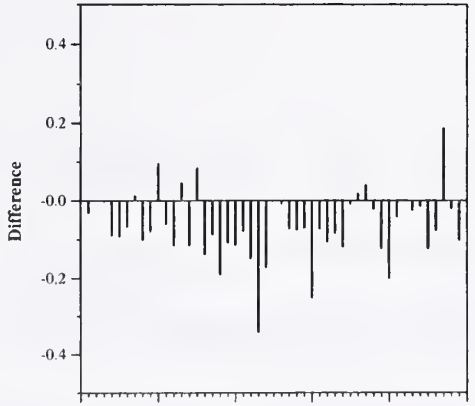


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		ntu1
Num topics		
	Relevant	New
Average precision	0.07	0.07
Average recall	0.47	0.07
Average F	0.115	0.063

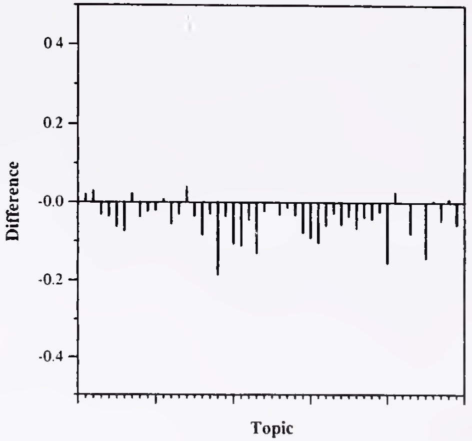


Difference from Median in F score per topic for relevant sentences

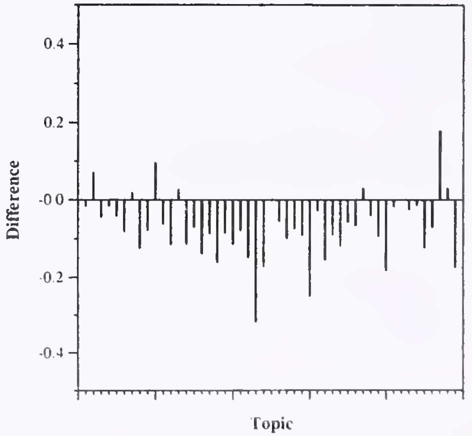


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		ntu2
Num topics		
	Relevant	New
Average precision	0.07	0.06
Average recall	0.47	0.07
Average F	0.110	0.059

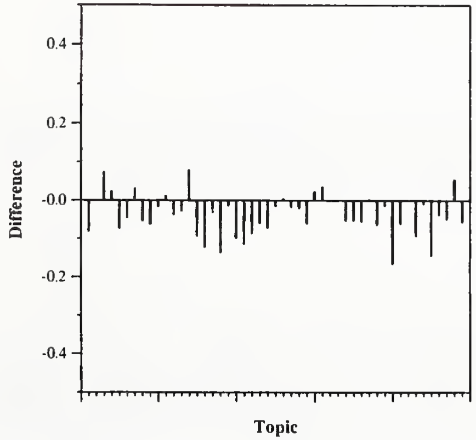


Difference from Median in F score per topic for relevant sentences

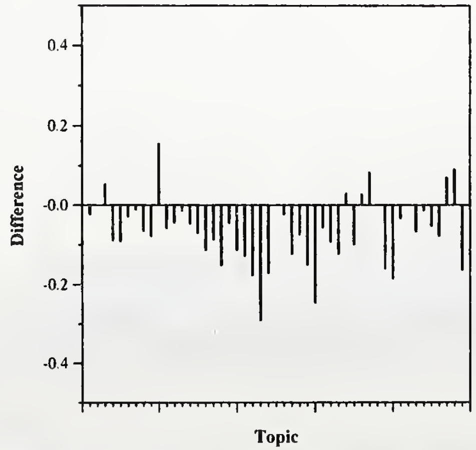


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		ntu3
Num topics		
	Relevant	New
Average precision	0.08	0.09
Average recall	0.40	0.06
Average F	0.116	0.065

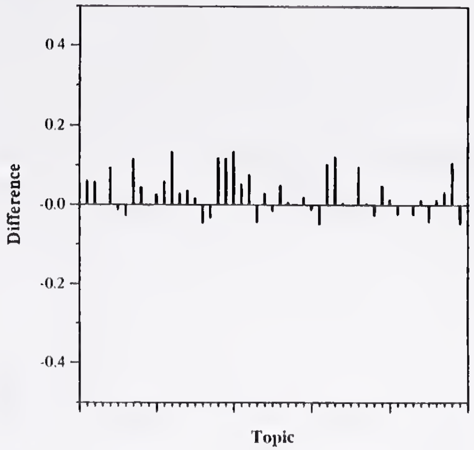


Difference from Median in F score per topic for relevant sentences

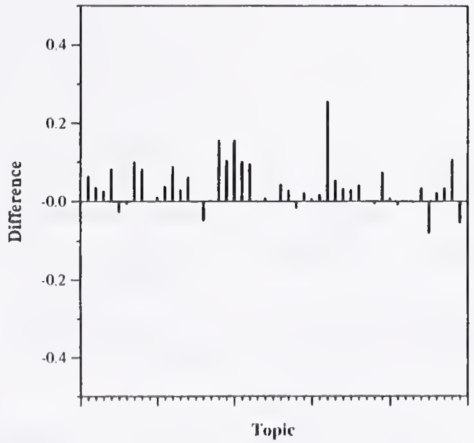


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		nttcslabnvp
Num topics		
	Relevant	New
Average precision	0.12	0.11
Average recall	0.54	0.44
Average F	0.183	0.165

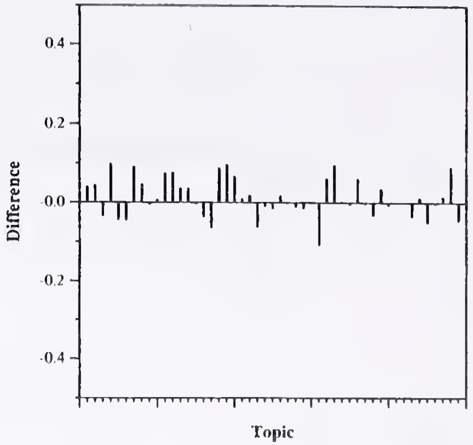


Difference from Median in F score per topic for relevant sentences

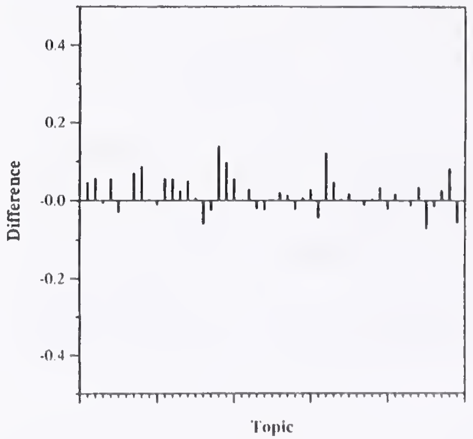


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		nttcslabnvr2
Num topics		
	Relevant	New
Average precision	0.10	0.09
Average recall	0.60	0.49
Average F	0.166	0.146

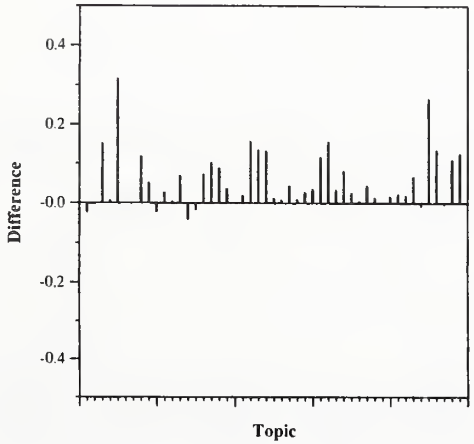


Difference from Median in F score per topic for relevant sentences

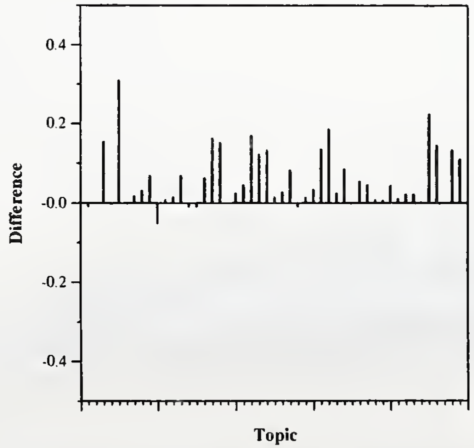


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		pircs2N01
Num topics		
	Relevant	New
Average precision	0.16	0.15
Average recall	0.49	0.39
Average F	0.209	0.188

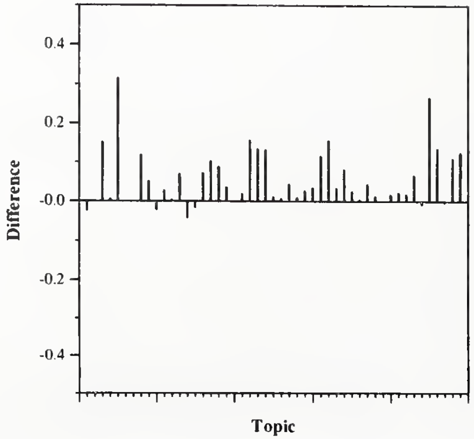


Difference from Median in F score per topic for relevant sentences

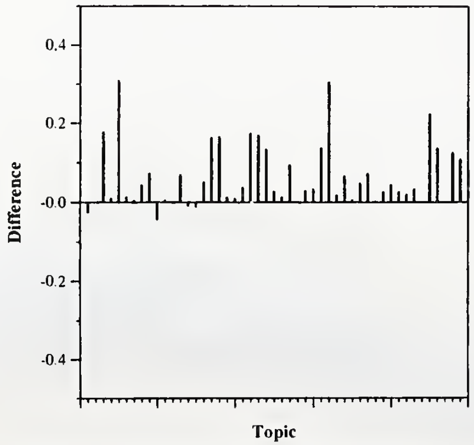


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		pircs2N02
Num topics		
	Relevant	New
Average precision	0.16	0.15
Average recall	0.49	0.43
Average F	0.209	0.193

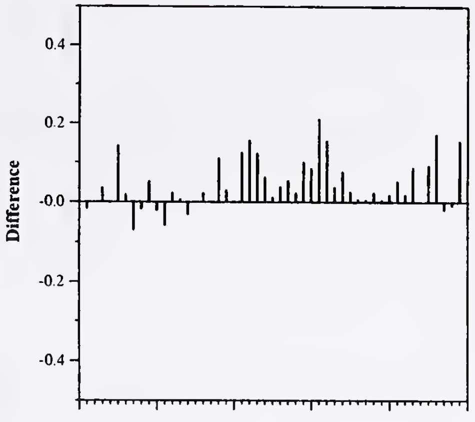


Difference from Median in F score per topic for relevant sentences

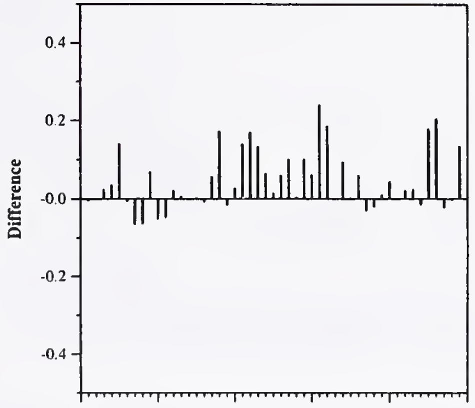


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		pircs2N03
Num topics		
	Relevant	New
Average precision	0.18	0.17
Average recall	0.40	0.31
Average F	0.197	0.175

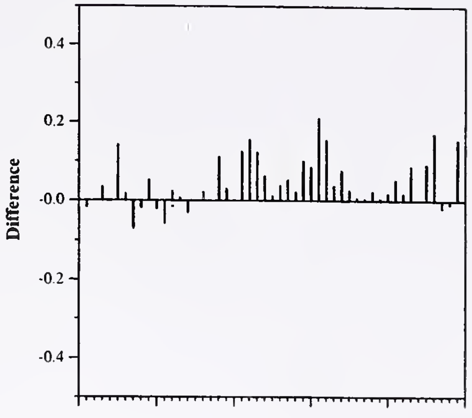


Difference from Median in F score per topic for relevant sentences

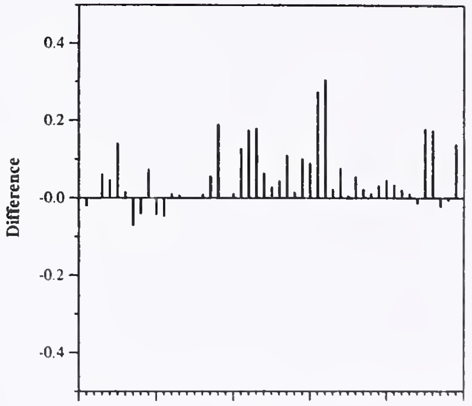


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		pircs2N04
Num topics		
	Relevant	New
Average precision	0.18	0.17
Average recall	0.40	0.36
Average F	0.197	0.184

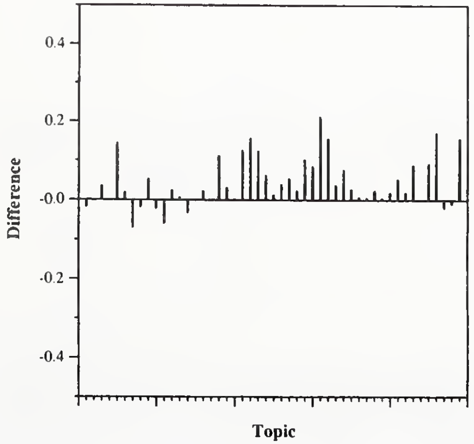


Difference from Median in F score per topic for relevant sentences

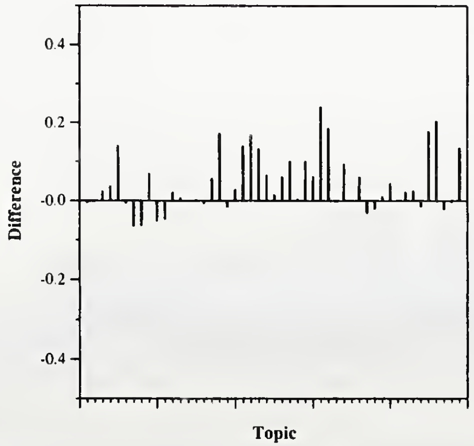


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		pircs2N05
Num topics		
	Relevant	New
Average precision	0.18	0.17
Average recall	0.40	0.31
Average F	0.197	0.175



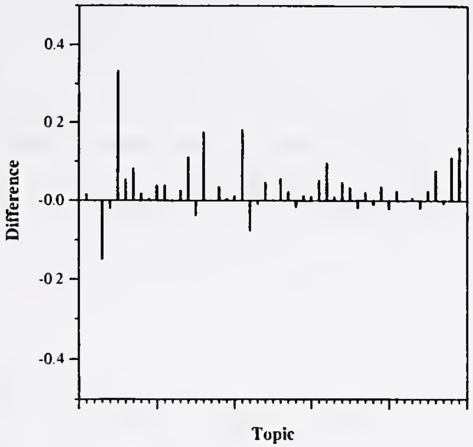
Difference from Median in F score per topic for relevant sentences



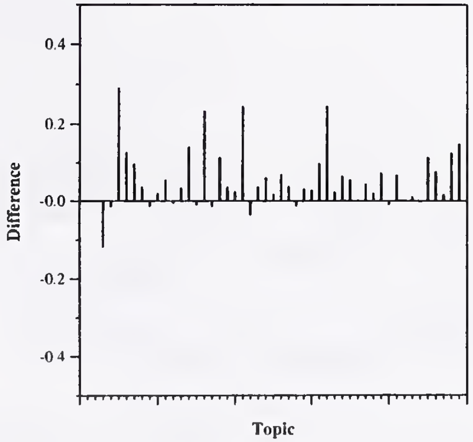
Difference from Median in F score per topic for new sentences

Summary Statistics	
Run ID	ss1
Num topics	

	Relevant	New
Average precision	0.14	0.14
Average recall	0.46	0.44
Average F	0.186	0.183



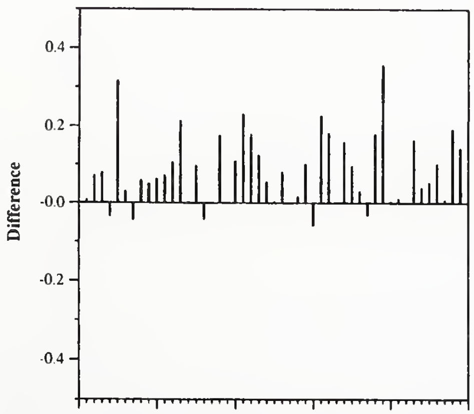
Difference from Median in F score per topic for relevant sentences



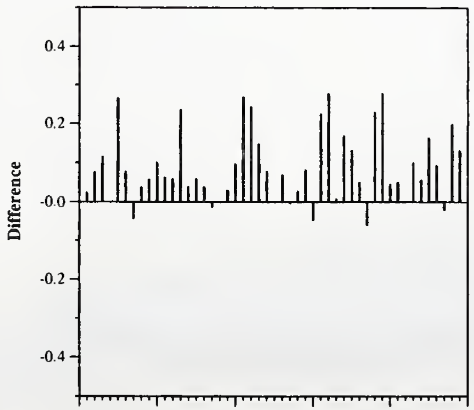
Difference from Median in F score per topic for new sentences

Summary Statistics	
Run ID	thunv1
Num topics	

	Relevant	New
Average precision	0.23	0.22
Average recall	0.34	0.30
Average F	0.235	0.217



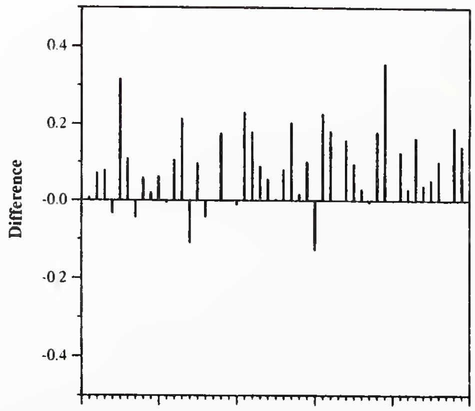
Difference from Median in F score per topic for relevant sentences



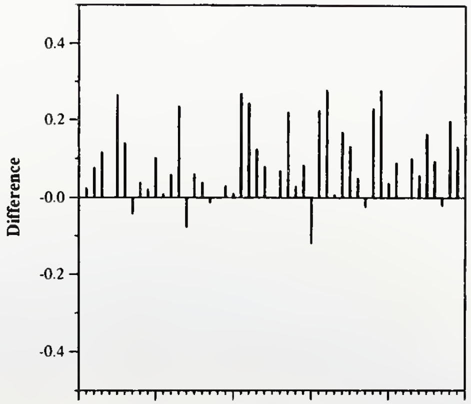
Difference from Median in F score per topic for new sentences

Summary Statistics	
Run ID	thunv2
Num topics	

	Relevant	New
Average precision	0.23	0.23
Average recall	0.34	0.29
Average F	0.235	0.216

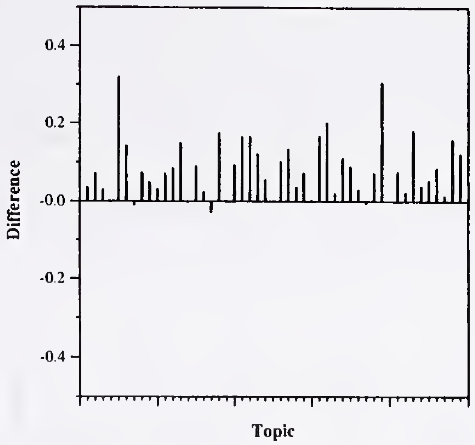


Difference from Median in F score per topic for relevant sentences

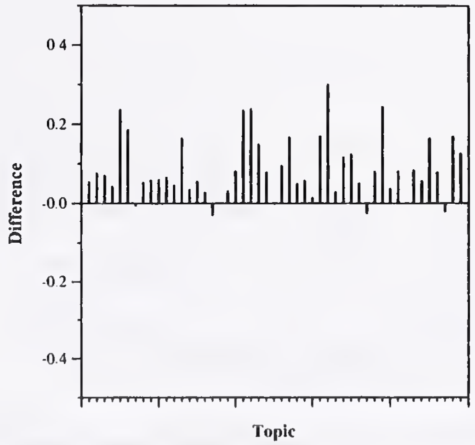


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		thunv3
Num topics		
	Relevant	New
Average precision	0.20	0.20
Average recall	0.41	0.35
Average F	0.235	0.216

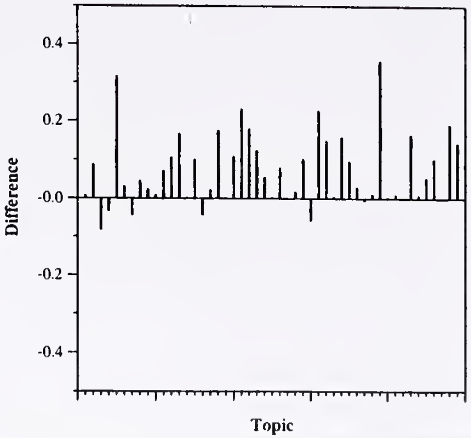


Difference from Median in F score per topic for relevant sentences

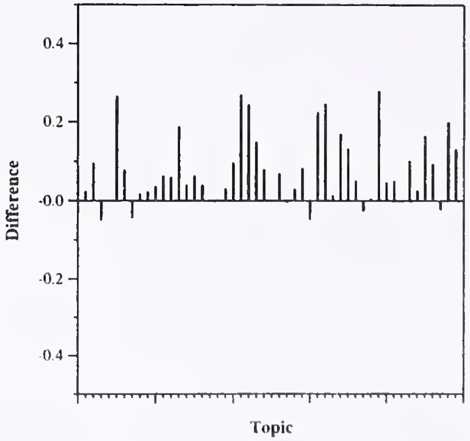


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		thunv4
Num topics		
	Relevant	New
Average precision	0.21	0.21
Average recall	0.33	0.28
Average F	0.225	0.206

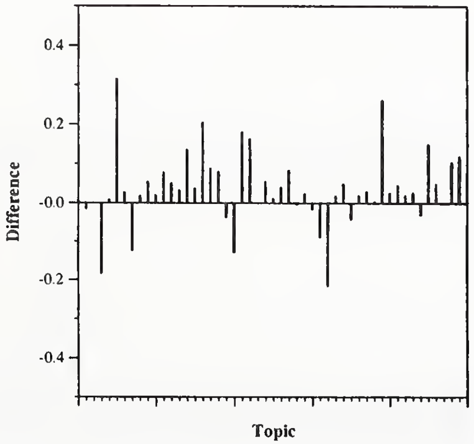


Difference from Median in F score per topic for relevant sentences

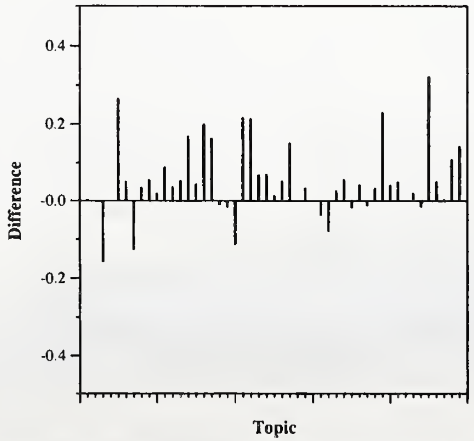


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		thunv5
Num topics		
	Relevant	New
Average precision	0.19	0.18
Average recall	0.35	0.31
Average F	0.190	0.181

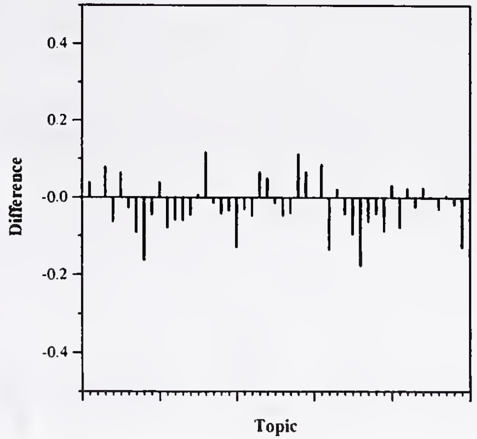


Difference from Median in F score per topic for relevant sentences

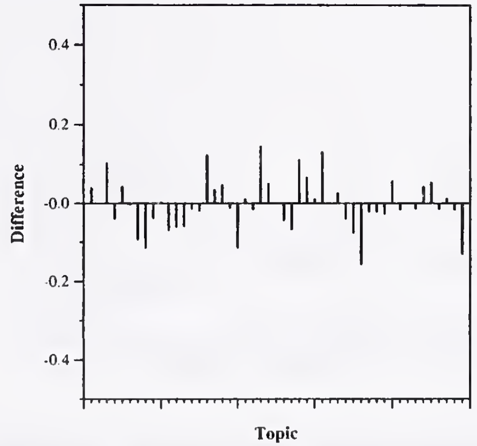


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID	UAmsT11ntcom	
Num topics		
	Relevant	New
Average precision	0.11	0.10
Average recall	0.23	0.22
Average F	0.131	0.125

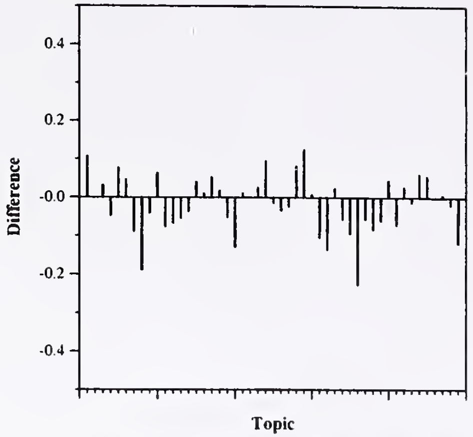


Difference from Median in F score per topic for relevant sentences

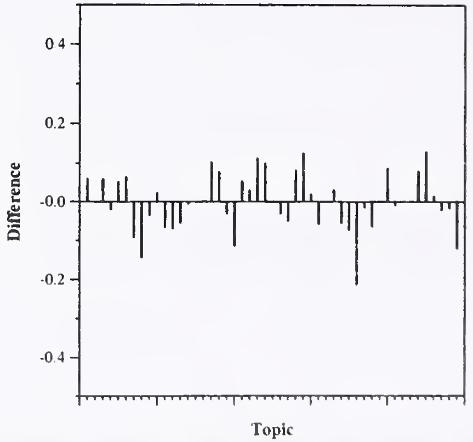


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID	UAmsT11ntlem	
Num topics		
	Relevant	New
Average precision	0.13	0.12
Average recall	0.19	0.18
Average F	0.136	0.128

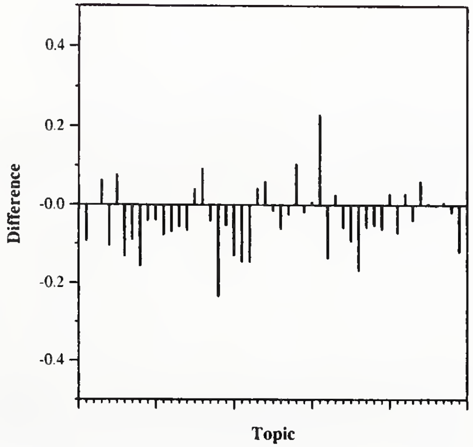


Difference from Median in F score per topic for relevant sentences

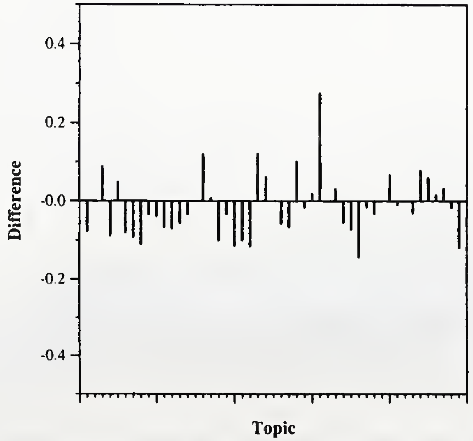


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID	UAmsT11ntste	
Num topics		
	Relevant	New
Average precision	0.11	0.10
Average recall	0.17	0.17
Average F	0.117	0.114

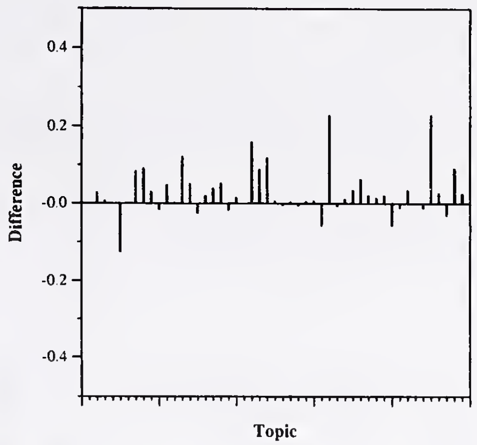


Difference from Median in F score per topic for relevant sentences

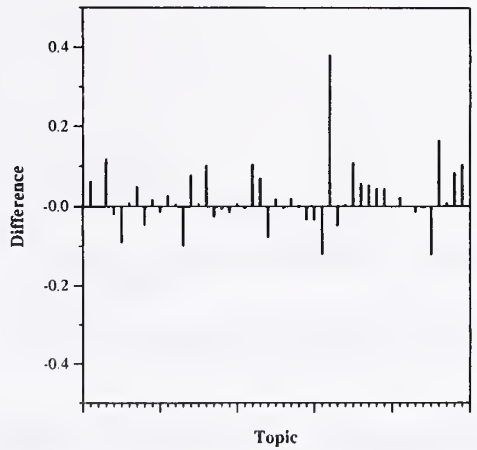


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID	UIowa02Nov4	
Num topics		
	Relevant	New
Average precision	0.12	0.12
Average recall	0.58	0.37
Average F	0.182	0.149

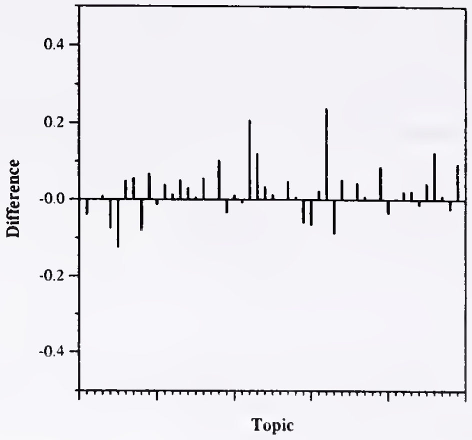


Difference from Median in F score per topic for relevant sentences

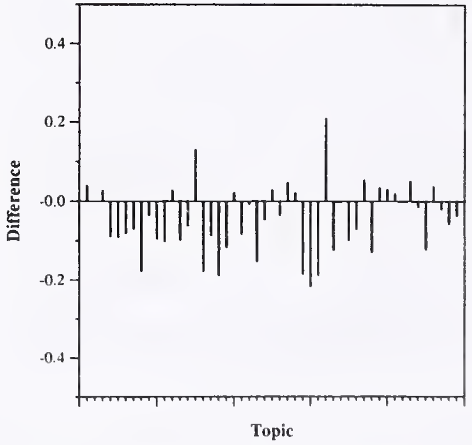


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID	UIowa02Nov5	
Num topics		
	Relevant	New
Average precision	0.14	0.14
Average recall	0.36	0.12
Average F	0.175	0.082

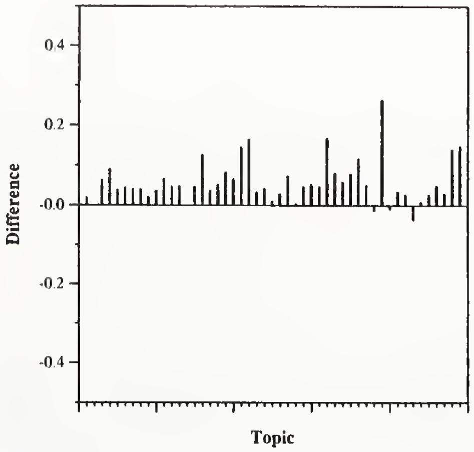


Difference from Median in F score per topic for relevant sentences

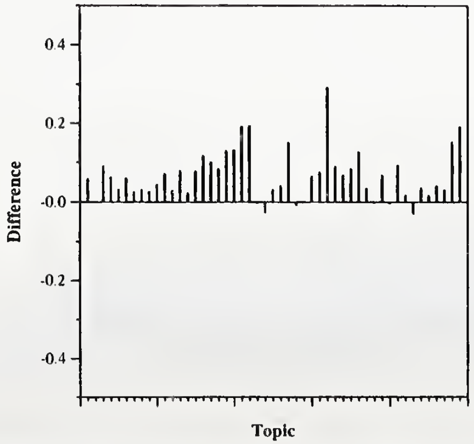


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		CIIR02tfkl
Num topics		
	Relevant	New
Average precision	0.14	0.14
Average recall	0.56	0.46
Average F	0.211	0.196

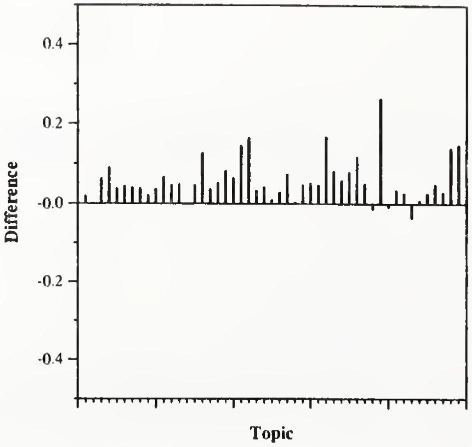


Difference from Median in F scorc pcr topic for relevant sentences

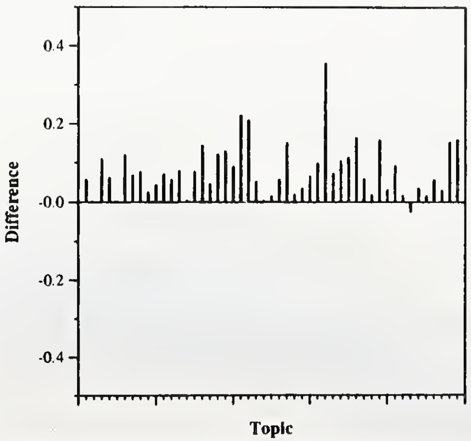


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		CIIR02tfnew
Num topics		
	Relevant	New
Average precision	0.14	0.14
Average recall	0.56	0.48
Average F	0.211	0.209

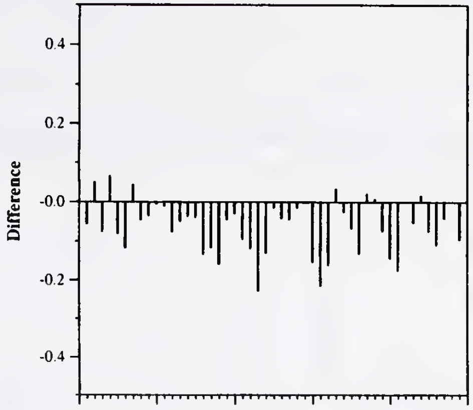


Difference from Median in F score per topic for relevant sentences

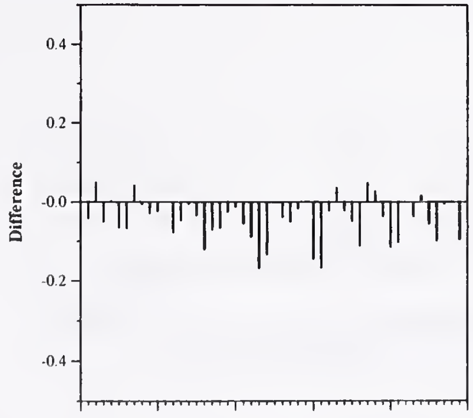


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		UMICH4
Num topics		
	Relevant	New
Average precision	0.06	0.05
Average recall	0.29	0.29
Average F	0.091	0.085

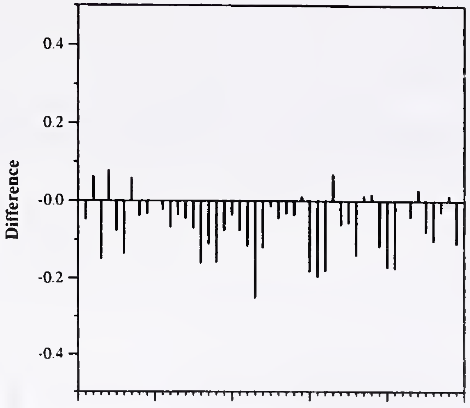


Difference from Median in F score per topic for relevant sentences

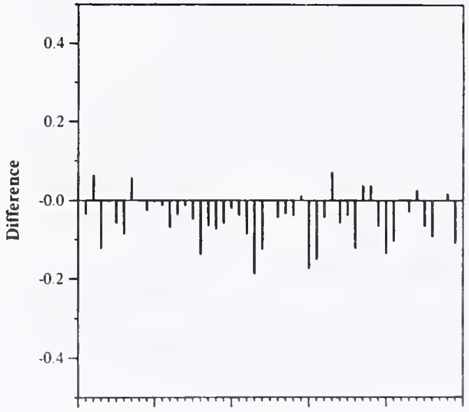


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		UMIch2
Num topics		
	Relevant	New
Average precision	0.06	0.06
Average recall	0.24	0.24
Average F	0.088	0.083

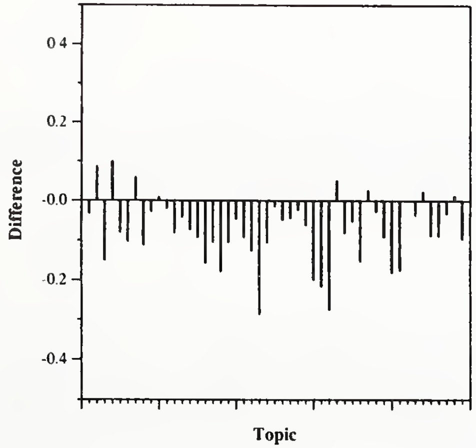


Difference from Median in F score per topic for relevant sentences

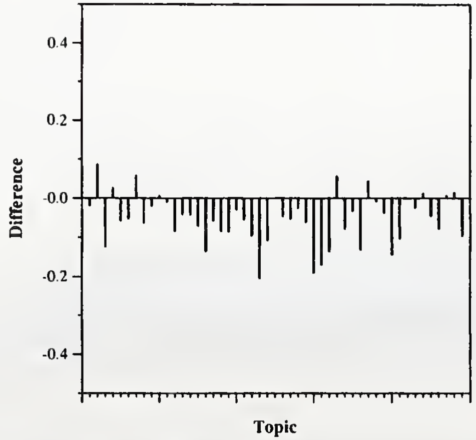


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		UMich3
Num topics		
	Relevant	New
Average precision	0.06	0.06
Average recall	0.17	0.17
Average F	0.080	0.076

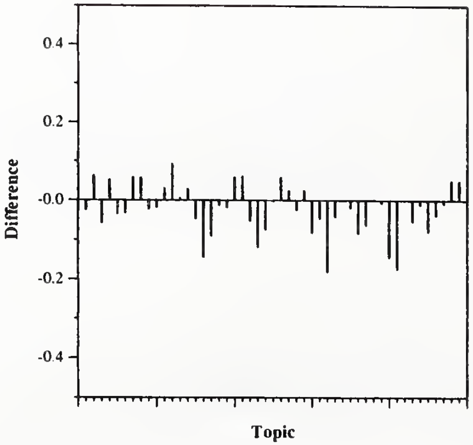


Difference from Median in F score per topic for relevant sentences

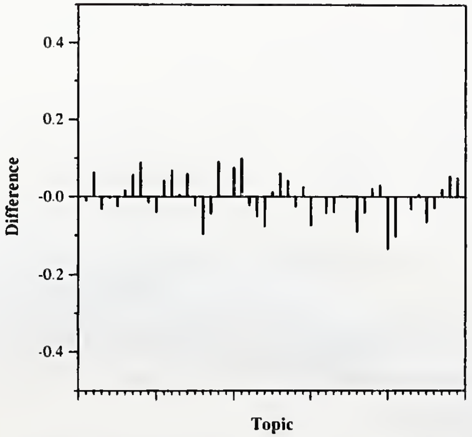


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		UMich5
Num topics		
	Relevant	New
Average precision	0.09	0.09
Average recall	0.35	0.35
Average F	0.132	0.126

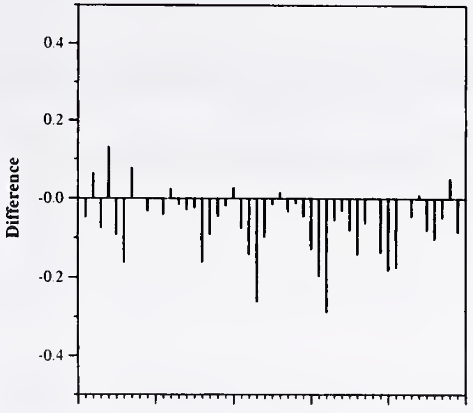


Difference from Median in F score per topic for relevant sentences

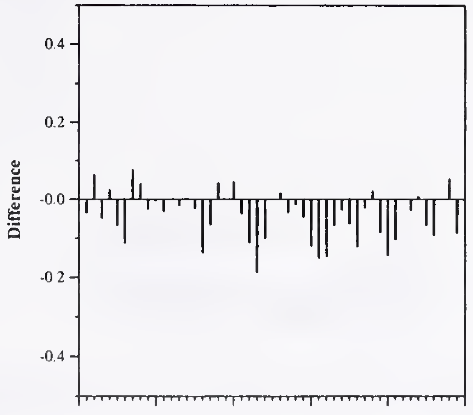


Difference from Median in F score per topic for new sentences

Summary Statistics		
Run ID		umich1
Num topics		
	Relevant	New
Average precision	0.06	0.06
Average recall	0.26	0.25
Average F	0.094	0.088

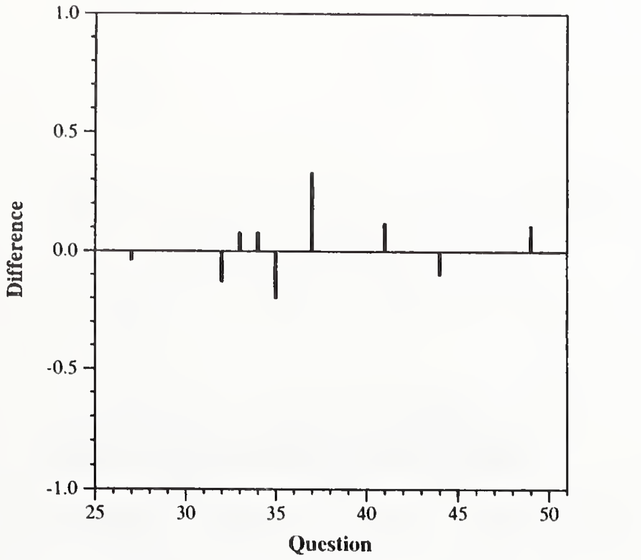


Difference from Median in F score per topic for relevant sentences



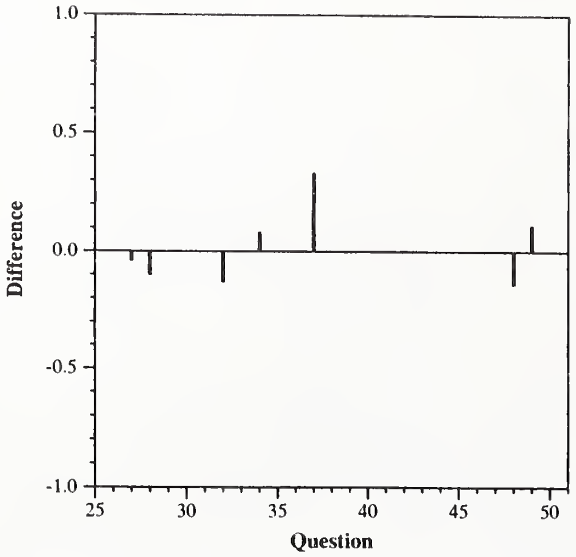
Difference from Median in F score per topic for new sentences

Summary Statistics	
Run ID	clr0211
Num questions	25
Average accuracy	0.06



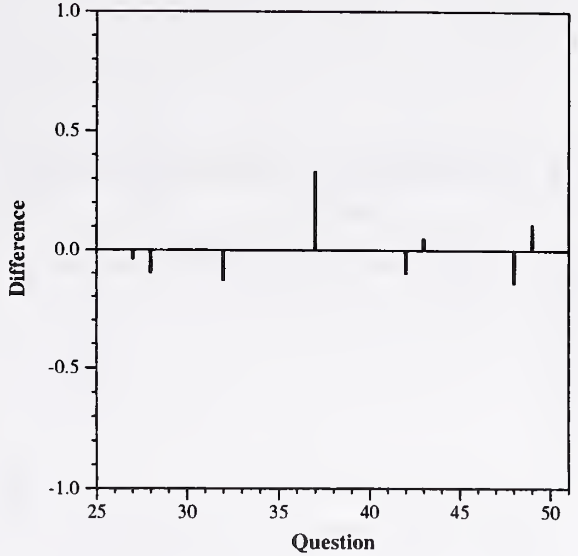
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	clr0212
Num questions	25
Average accuracy	0.05



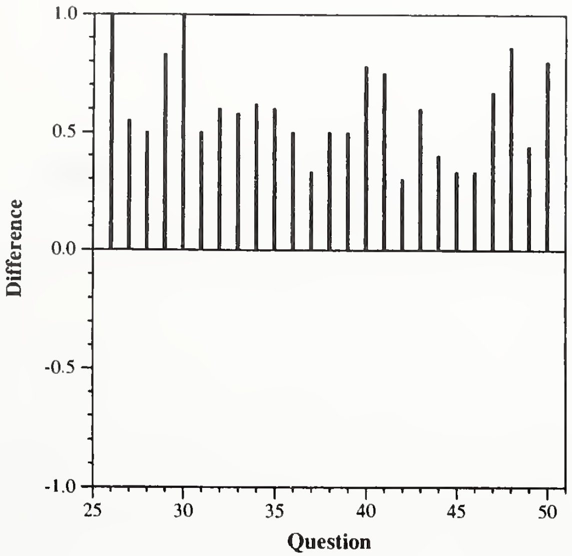
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	clr0213
Num questions	25
Average accuracy	0.05



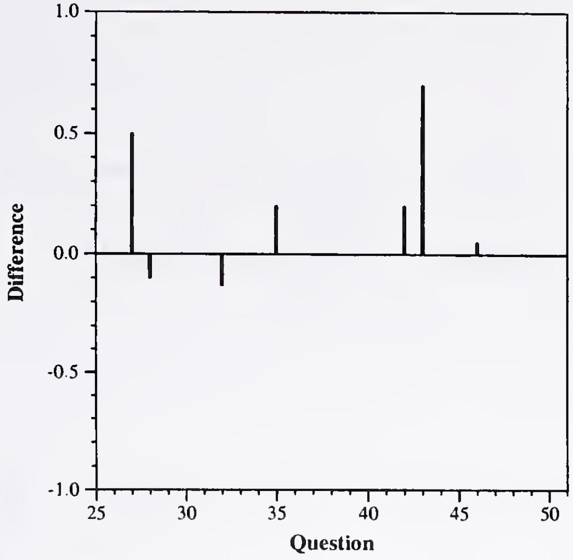
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	LCClist2002
Num questions	25
Average accuracy	0.65



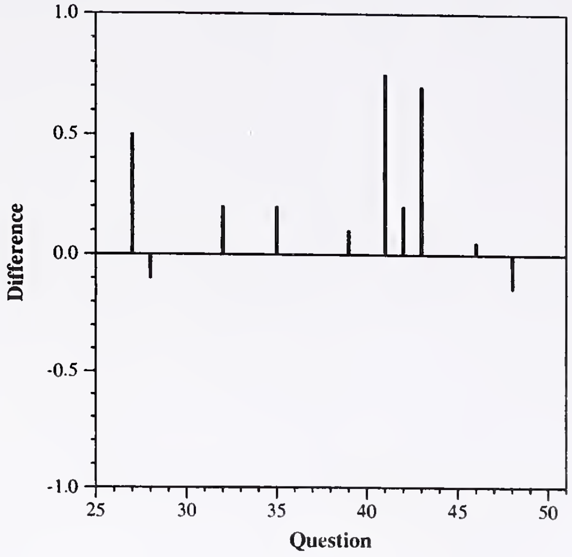
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	SUT11IR1LT
Num questions	25
Average accuracy	0.11



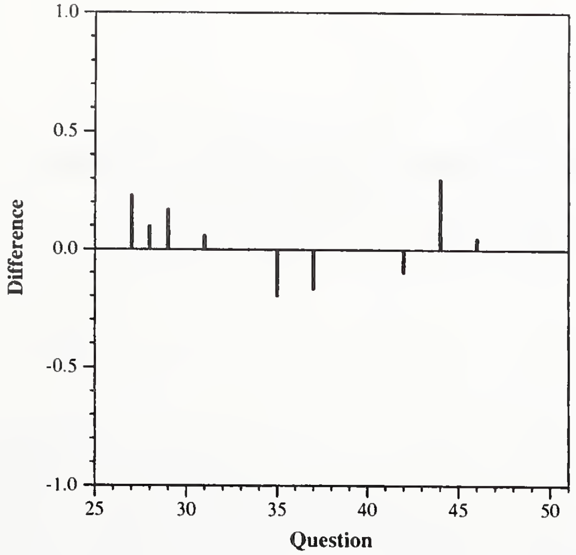
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	SUT11IR1LT2
Num questions	25
Average accuracy	0.15



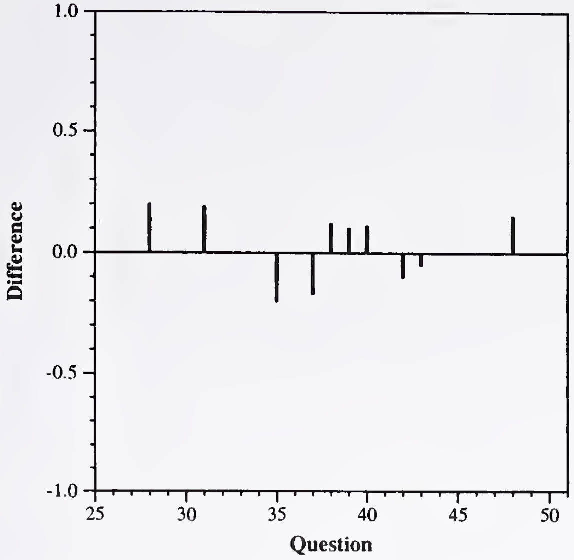
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	UdeMlistNoW
Num questions	25
Average accuracy	0.07



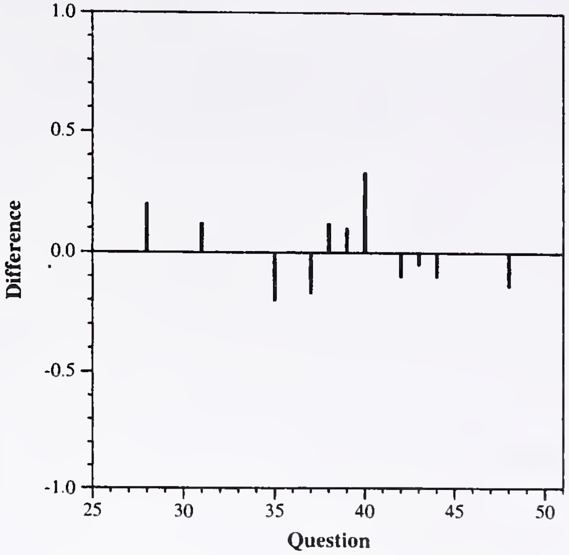
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	sheft11lo
Num questions	25
Average accuracy	0.06



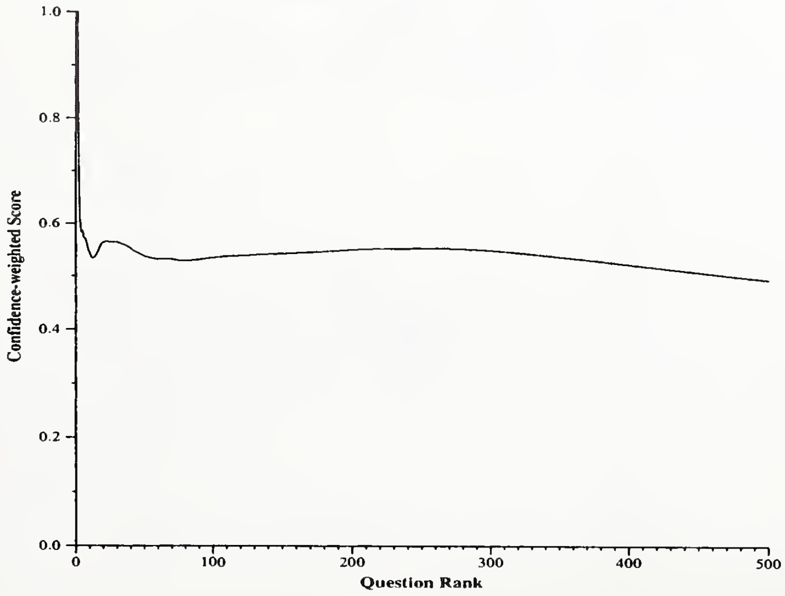
Difference from Median in average accuracy per question

Summary Statistics	
Run ID	sheft11log
Num questions	25
Average accuracy	0.06

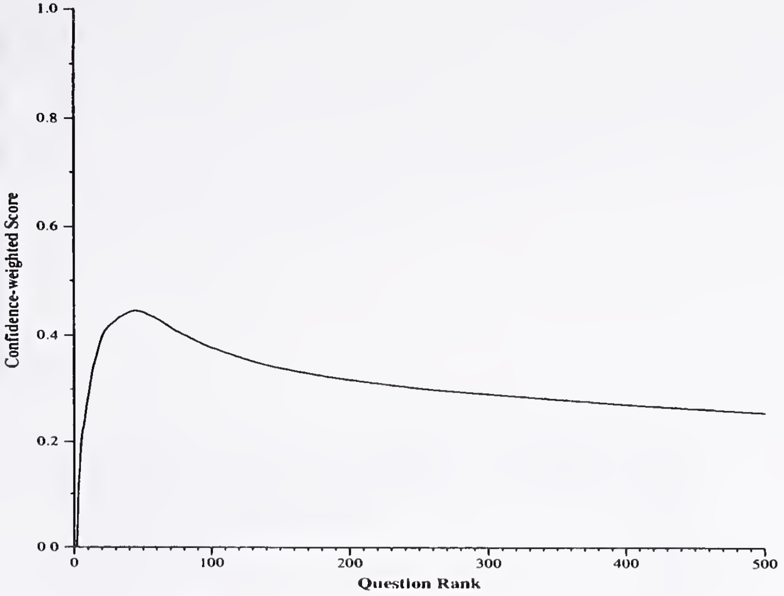


Difference from Median in average accuracy per question

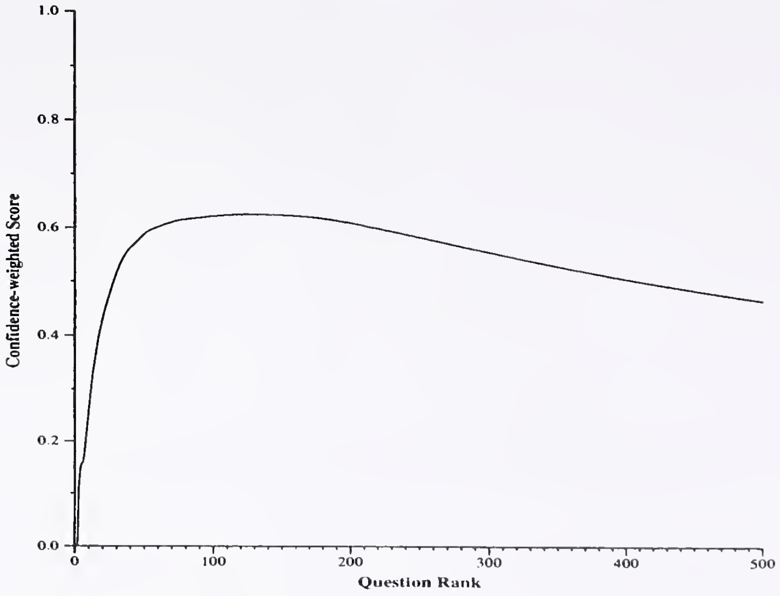
Summary Statistics	
Run ID	ali2002b
Num questions	500
Number wrong	302
Number unsupported	2
Number inexact	15
Number right	181
Confidence-weighted score	0.496
Precision of recognizing no answer	39 / 250 = 0.156
Recall of recognizing no answer	39 / 46 = 0.848



Summary Statistics	
Run ID	BBN2002A
Num questions	500
Number wrong	390
Number unsupported	5
Number inexact	12
Number right	93
Confidence-weighted score	0.257
Precision of recognizing no answer	4 / 22 = 0.182
Recall of recognizing no answer	4 / 46 = 0.087

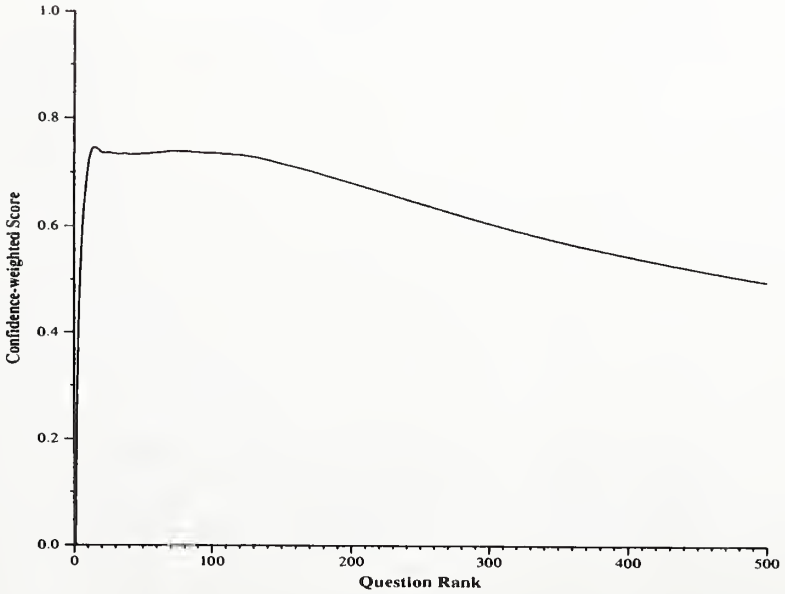


Summary Statistics	
Run ID	BBN2002B
Num questions	500
Number wrong	310
Number unsupported	30
Number inexact	16
Number right	144
Confidence-weighted score	0.468
Precision of recognizing no answer	4 / 22 = 0.182
Recall of recognizing no answer	4 / 46 = 0.087

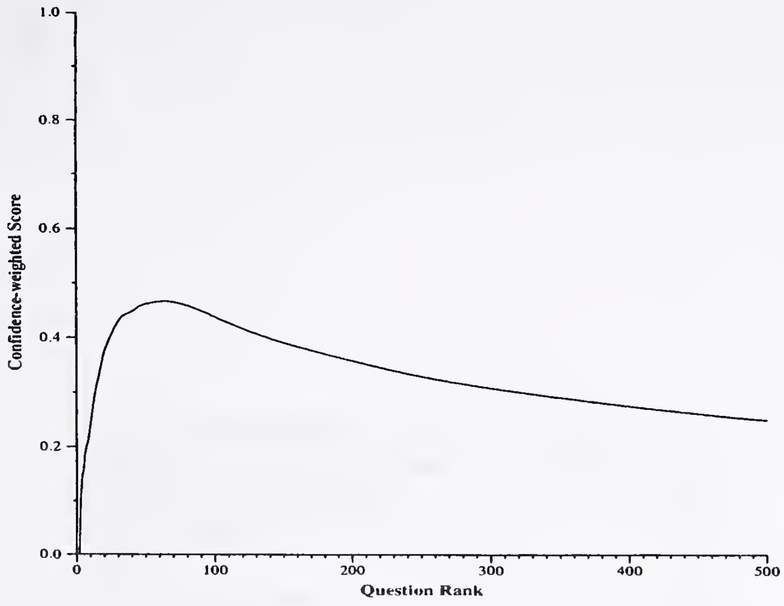


Question Answering track, Main task results — BBN

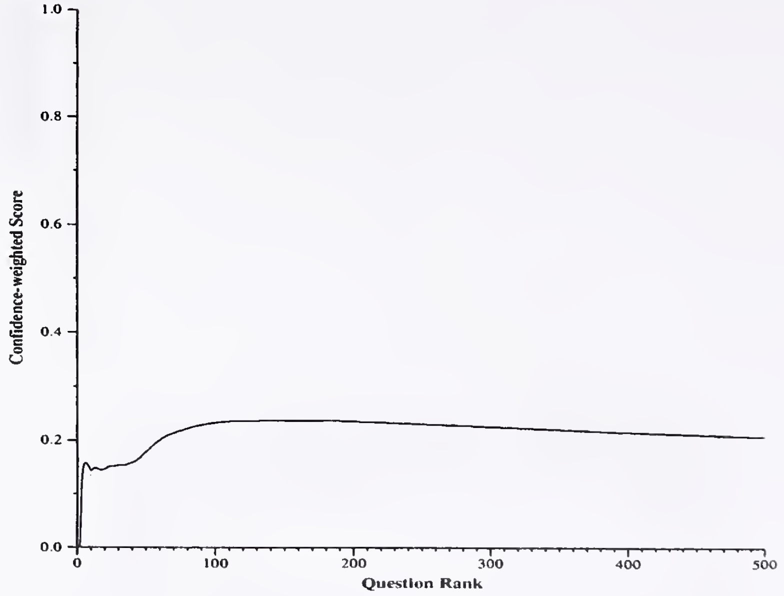
Summary Statistics	
Run ID	BBN2002C
Num questions	500
Number wrong	313
Number unsupported	27
Number inexact	18
Number right	142
Confidence-weighted score	0.499
Precision of recognizing no answer	4 / 22 = 0.182
Recall of recognizing no answer	4 / 46 = 0.087



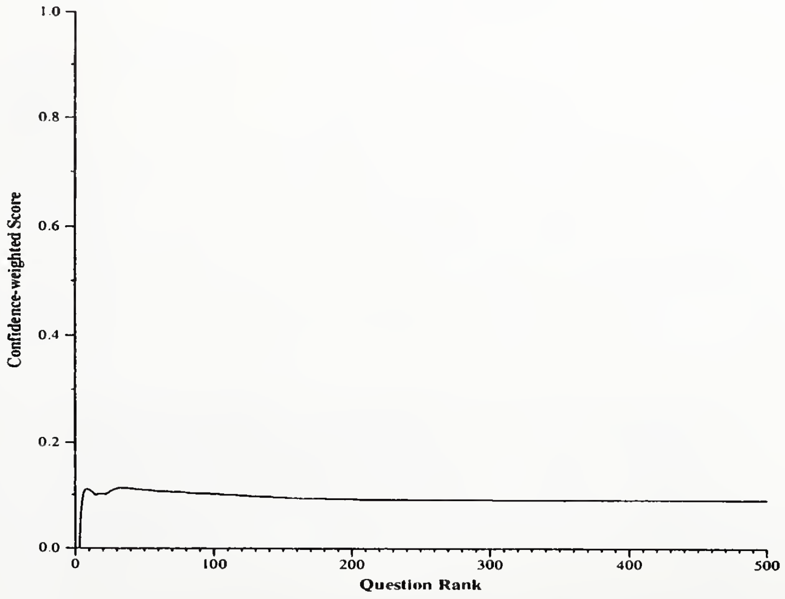
Summary Statistics	
Run ID	CMUJAV000495
Num questions	500
Number wrong	402
Number unsupported	10
Number inexact	13
Number right	75
Confidence-weighted score	0.251
Precision of recognizing no answer	12 / 79 = 0.152
Recall of recognizing no answer	12 / 46 = 0.261



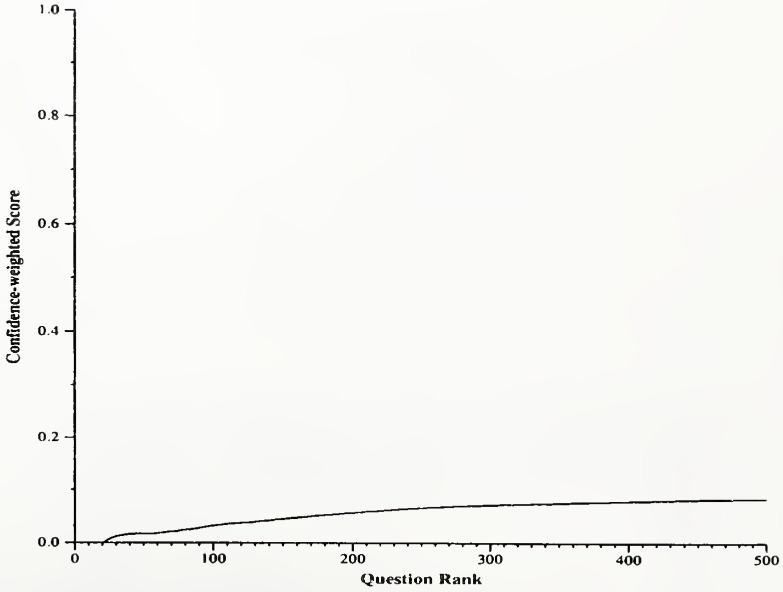
Summary Statistics	
Run ID	CMUJAV000501
Num questions	500
Number wrong	394
Number unsupported	8
Number inexact	12
Number right	86
Confidence-weighted score	0.209
Precision of recognizing no answer	10 / 61 = 0.164
Recall of recognizing no answer	10 / 46 = 0.217



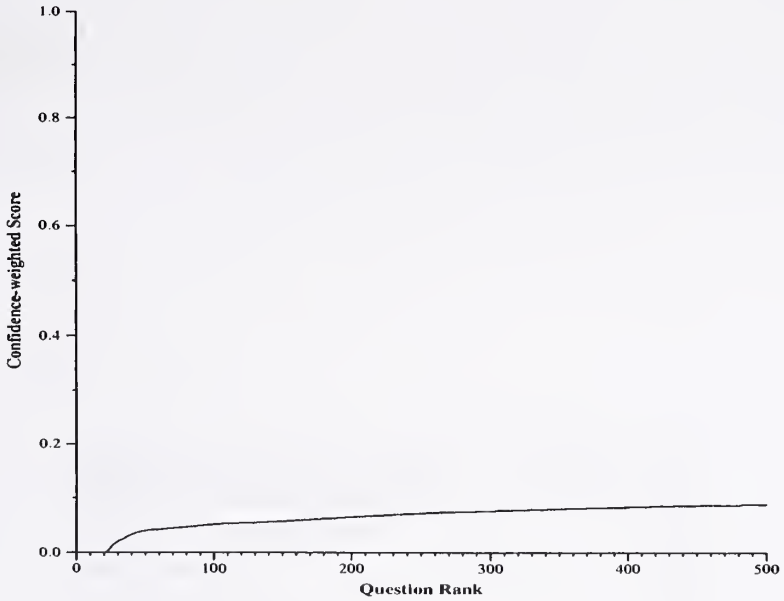
Summary Statistics	
Run ID	ICTQA11a
Num questions	500
Number wrong	445
Number unsupported	9
Number inexact	4
Number right	42
Confidence-weighted score	0.091
Precision of recognizing no answer	10 / 58 = 0.172
Recall of recognizing no answer	10 / 46 = 0.217



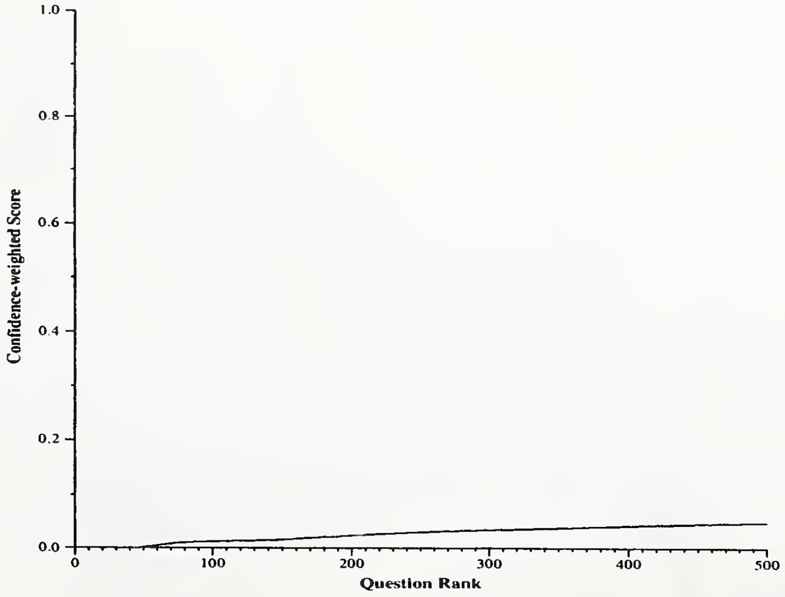
Summary Statistics	
Run ID	ICTQA11b
Num questions	500
Number wrong	440
Number unsupported	7
Number inexact	6
Number right	47
Confidence-weighted score	0.084
Precision of recognizing no answer	9 / 69 = 0.130
Recall of recognizing no answer	9 / 46 = 0.196



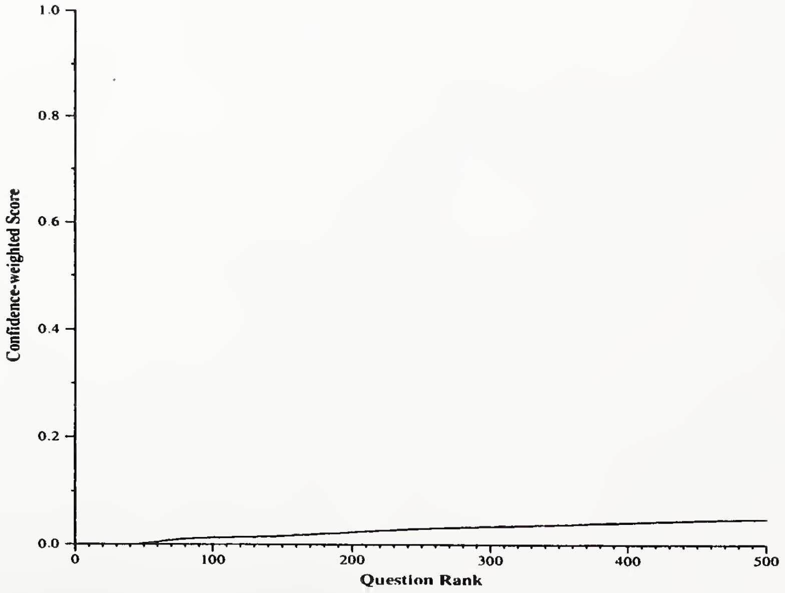
Summary Statistics	
Run ID	ICTQA11c
Num questions	500
Number wrong	435
Number unsupported	8
Number inexact	9
Number right	48
Confidence-weighted score	0.088
Precision of recognizing no answer	9 / 47 = 0.191
Recall of recognizing no answer	9 / 46 = 0.196



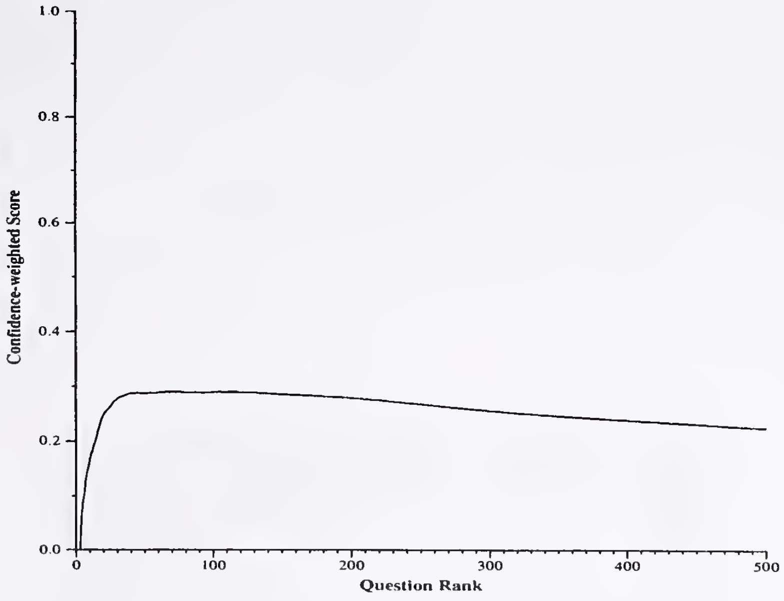
Summary Statistics	
Run ID	clr02b1
Num questions	500
Number wrong	452
Number unsupported	2
Number inexact	10
Number right	36
Confidence-weighted score	0.049
Precision of recognizing no answer	0 / 8 = 0.000
Recall of recognizing no answer	0 / 46 = 0.000



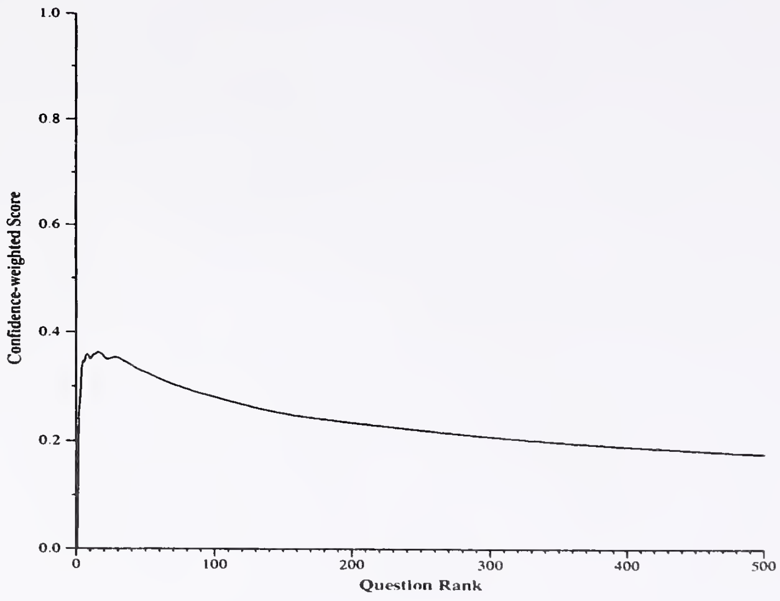
Summary Statistics	
Run ID	clr02b2
Num questions	500
Number wrong	452
Number unsupported	2
Number inexact	10
Number right	36
Confidence-weighted score	0.049
Precision of recognizing no answer	0 / 8 = 0.000
Recall of recognizing no answer	0 / 46 = 0.000



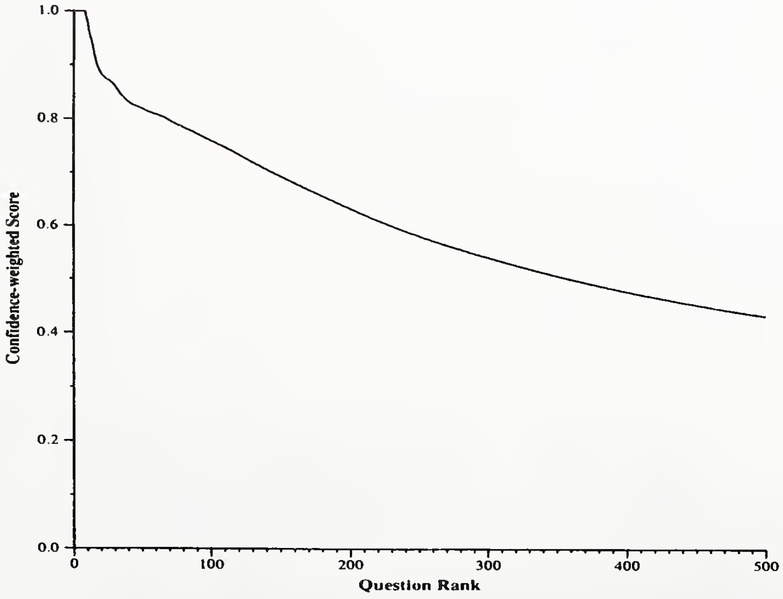
Summary Statistics	
Run ID	2002cuaq1
Num questions	500
Number wrong	411
Number unsupported	7
Number inexact	3
Number right	79
Confidence-weighted score	0.226
Precision of recognizing no answer	0 / 9 = 0.000
Recall of recognizing no answer	0 / 46 = 0.000



Summary Statistics	
Run ID	2002cuaq2
Num questions	500
Number wrong	432
Number unsupported	8
Number inexact	2
Number right	58
Confidence-weighted score	0.178
Precision of recognizing no answer	15 / 195 = 0.077
Recall of recognizing no answer	15 / 46 = 0.326

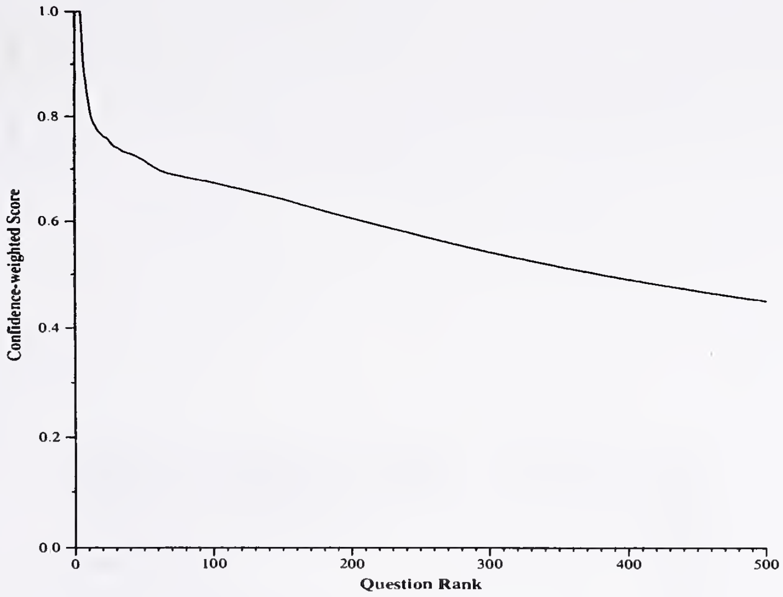


Summary Statistics	
Run ID	FDUT11QA1
Num questions	500
Number wrong	369
Number unsupported	1
Number inexact	6
Number right	124
Confidence-weighted score	0.434
Precision of recognizing no answer	44 / 316 = 0.139
Recall of recognizing no answer	44 / 46 = 0.957

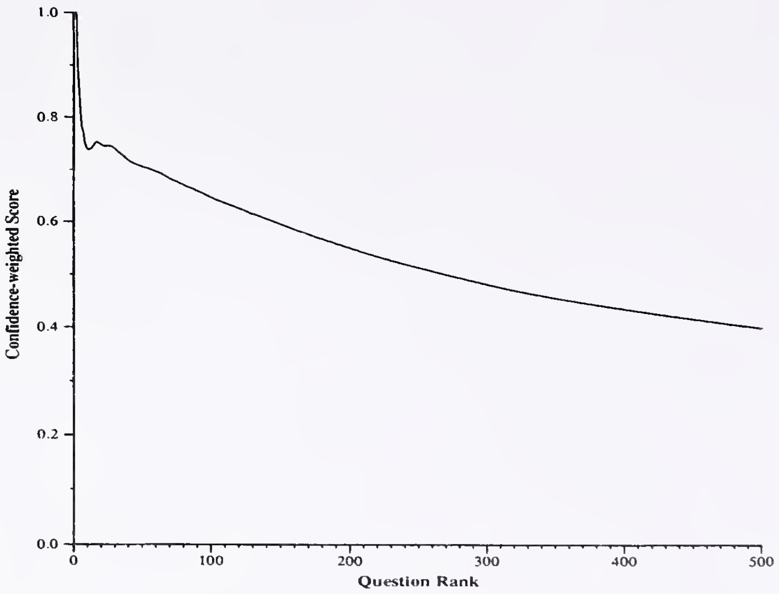


Question Answering track, Main task results — IBM T.J. Watson Research Center-Ittycheriah

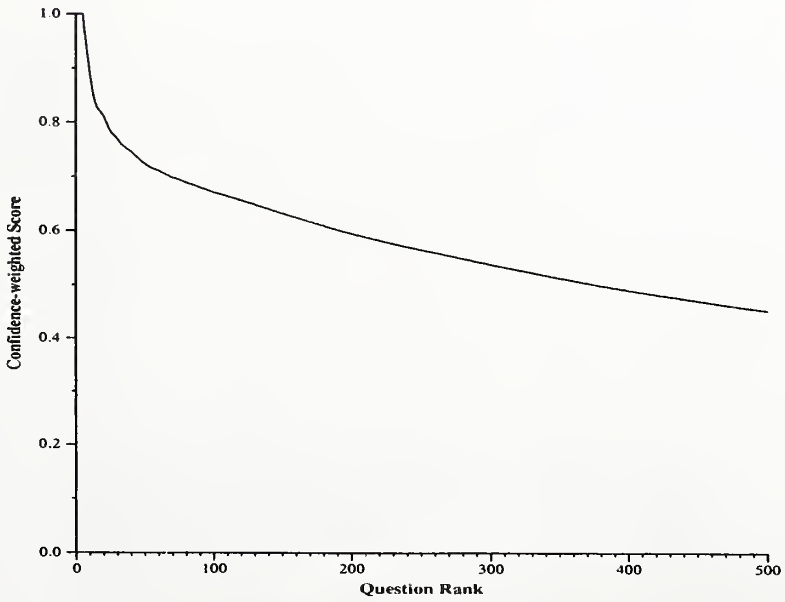
Summary Statistics	
Run ID	ibmsqa02a
Num questions	500
Number wrong	312
Number unsupported	11
Number inexact	37
Number right	140
Confidence-weighted score	0.454
Precision of recognizing no answer	12 / 83 = 0.145
Recall of recognizing no answer	12 / 46 = 0.261



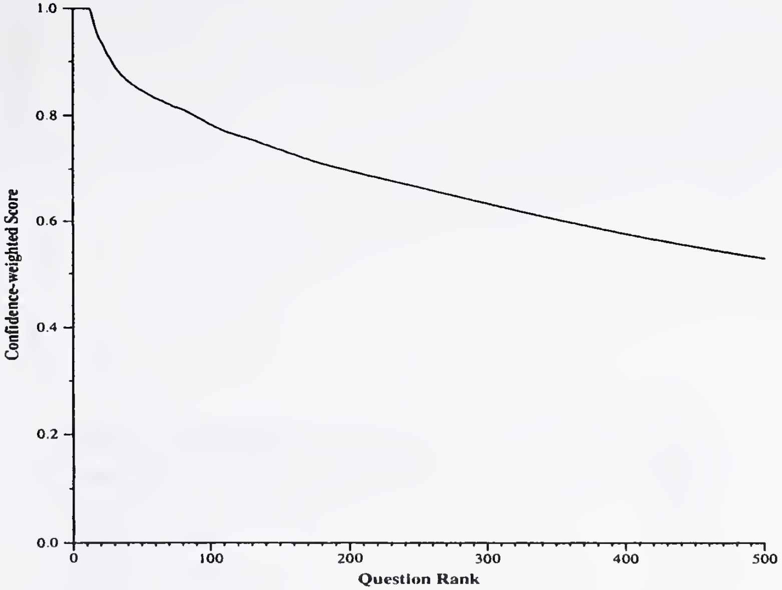
Summary Statistics	
Run ID	ibmsqa02b
Num questions	500
Number wrong	328
Number unsupported	8
Number inexact	43
Number right	121
Confidence-weighted score	0.403
Precision of recognizing no answer	0 / 0 = 0.000
Recall of recognizing no answer	0 / 0 = 0.000



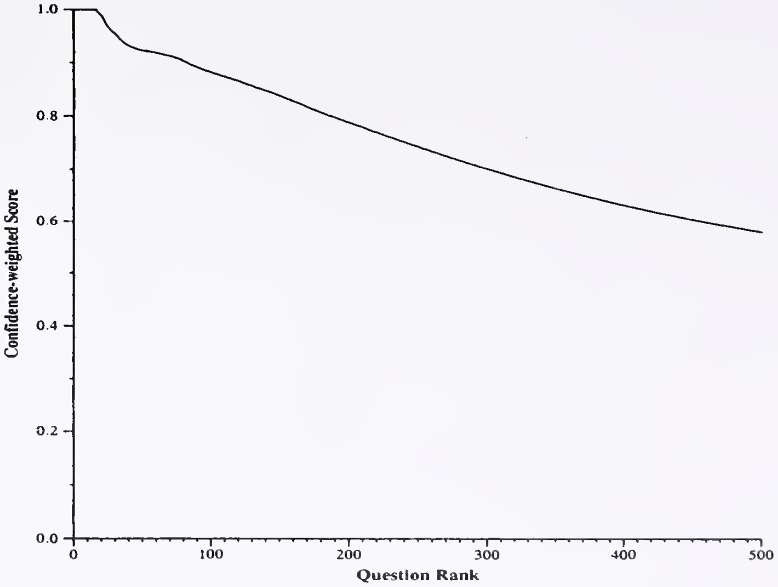
Summary Statistics	
Run ID	ibmsqa02c
Num questions	500
Number wrong	293
Number unsupported	18
Number inexact	44
Number right	145
Confidence-weighted score	0.455
Precision of recognizing no answer	11 / 49 = 0.224
Recall of recognizing no answer	11 / 46 = 0.239



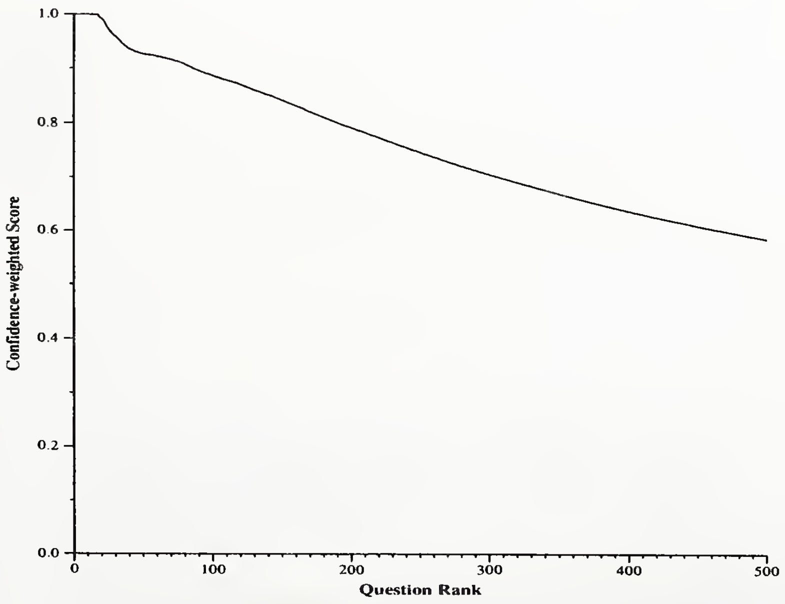
Summary Statistics	
Run ID	IBMPQ
Num questions	500
Number wrong	306
Number unsupported	14
Number inexact	11
Number right	169
Confidence-weighted score	0.534
Precision of recognizing no answer	33 / 201 = 0.164
Recall of recognizing no answer	33 / 46 = 0.717



Summary Statistics	
Run ID	IBMPQSQA
Num questions	500
Number wrong	300
Number unsupported	13
Number inexact	9
Number right	178
Confidence-weighted score	0.586
Precision of recognizing no answer	29 / 148 = 0.196
Recall of recognizing no answer	29 / 46 = 0.630

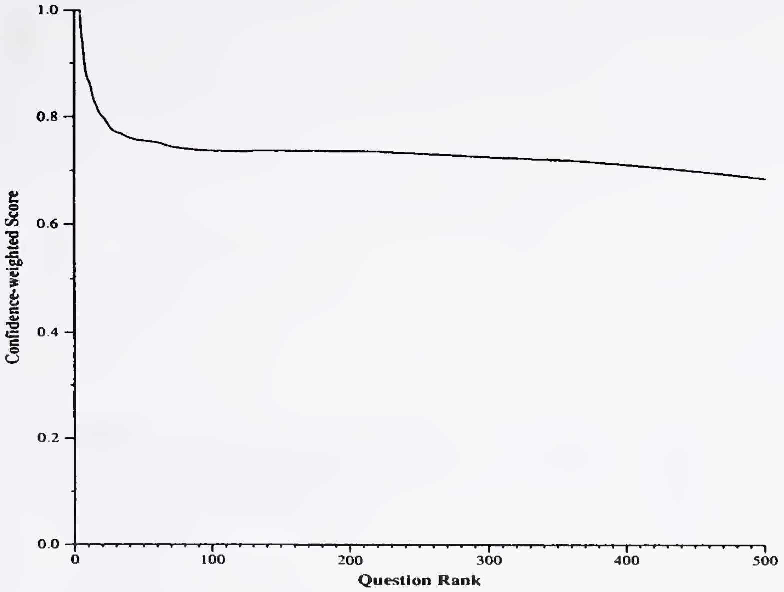


Summary Statistics	
Run ID	IBMPQSQACYC
Num questions	500
Number wrong	299
Number unsupported	13
Number inexact	9
Number right	179
Confidence-weighted score	0.588
Precision of recognizing no answer	29 / 148 = 0.196
Recall of recognizing no answer	29 / 46 = 0.630

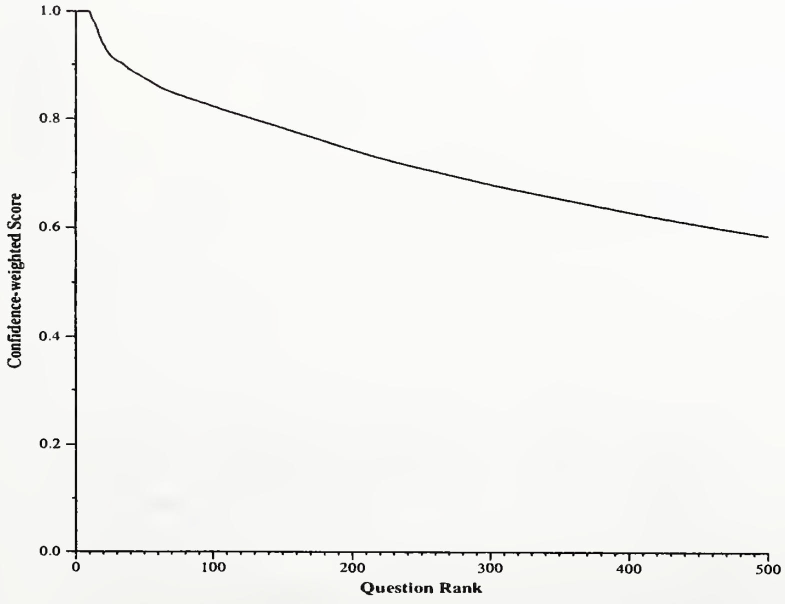


Question Answering track, Main task results — InsightSoft-M

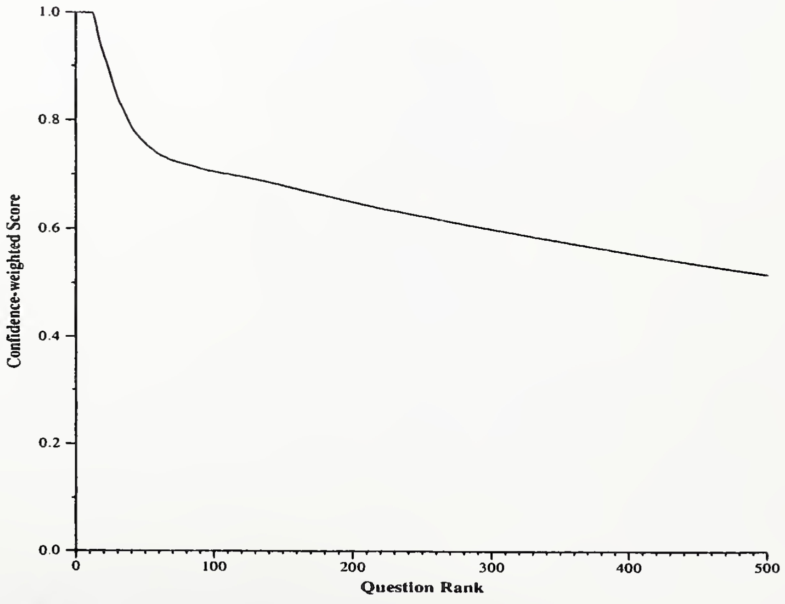
Summary Statistics	
Run ID	exactanswer
Num questions	500
Number wrong	209
Number unsupported	8
Number inexact	12
Number right	271
Confidence-weighted score	0.691
Precision of recognizing no answer	39 / 176 = 0.222
Recall of recognizing no answer	39 / 46 = 0.848



Summary Statistics	
Run ID	IRST02D1
Num questions	500
Number wrong	267
Number unsupported	24
Number inexact	17
Number right	192
Confidence-weighted score	0.589
Precision of recognizing no answer	10 / 60 = 0.167
Recall of recognizing no answer	10 / 46 = 0.217

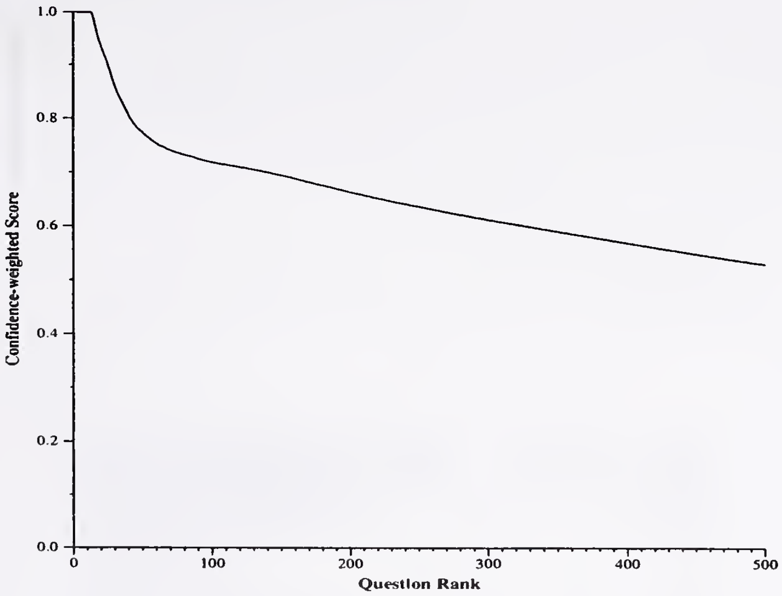


Summary Statistics	
Run ID	IRST02D2
Num questions	500
Number wrong	294
Number unsupported	19
Number inexact	14
Number right	173
Confidence-weighted score	0.520
Precision of recognizing no answer	14 / 71 = 0.197
Recall of recognizing no answer	14 / 46 = 0.304

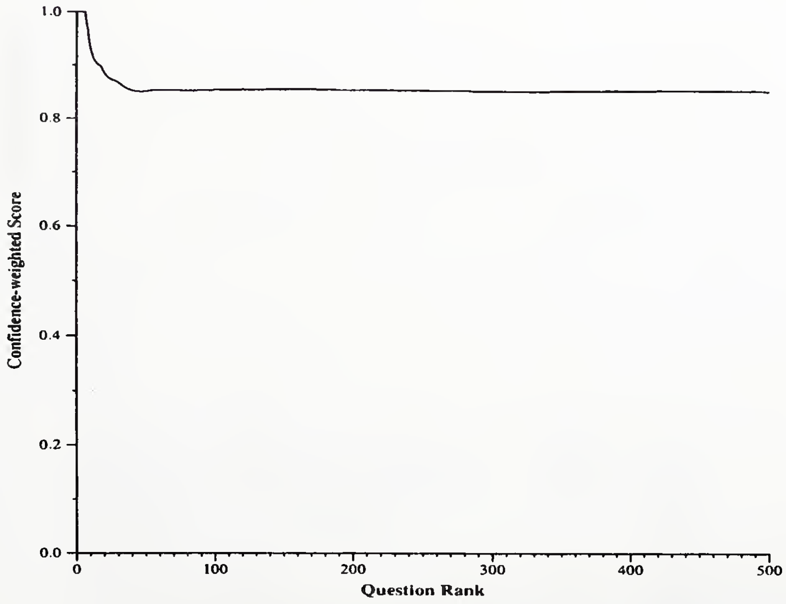


Question Answering track, Main task results — ITC-irst

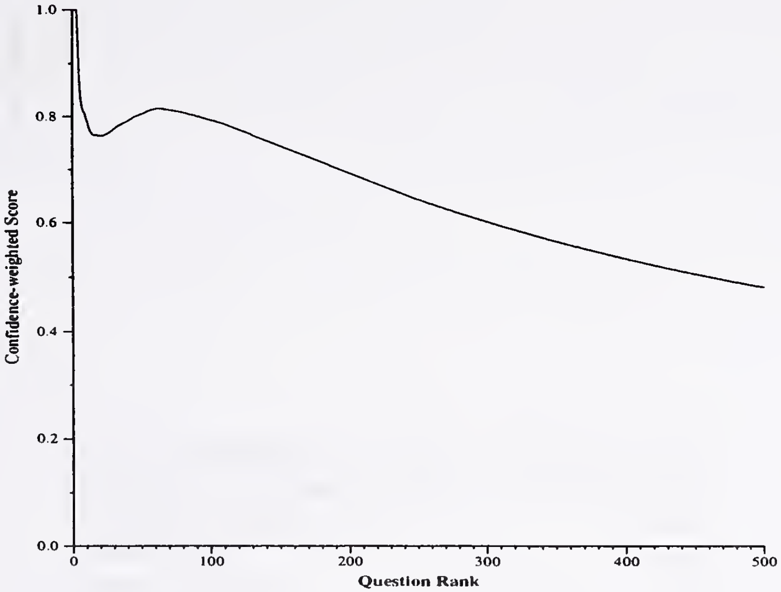
Summary Statistics	
Run ID	IRST02D3
Num questions	500
Number wrong	284
Number unsupported	23
Number inexact	16
Number right	177
Confidence-weighted score	0.533
Precision of recognizing no answer	8 / 30 = 0.267
Recall of recognizing no answer	8 / 46 = 0.174



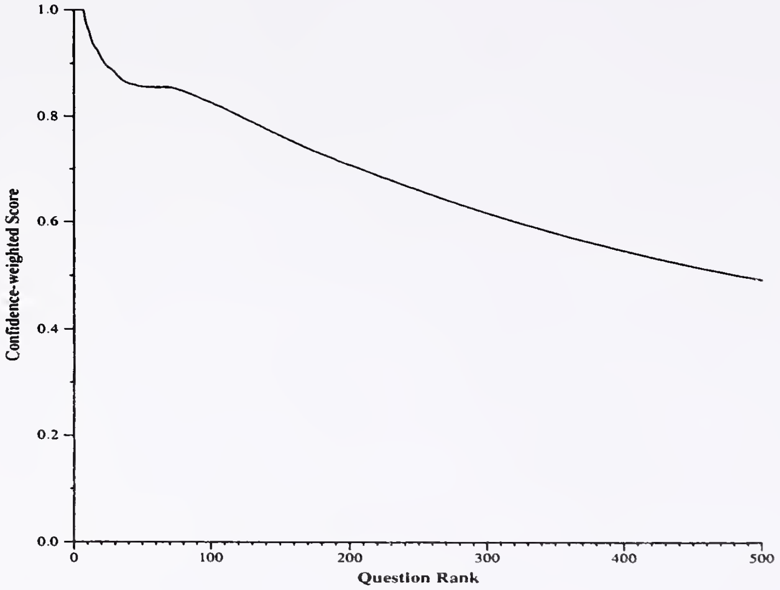
Summary Statistics	
Run ID	LCCmain2002
Num questions	500
Number wrong	63
Number unsupported	14
Number inexact	8
Number right	415
Confidence-weighted score	0.856
Precision of recognizing no answer	37 / 64 = 0.578
Recall of recognizing no answer	37 / 46 = 0.804



Summary Statistics	
Run ID	limsiQalir1
Num questions	500
Number wrong	342
Number unsupported	21
Number inexact	7
Number right	130
Confidence-weighted score	0.485
Precision of recognizing no answer	9 / 43 = 0.209
Recall of recognizing no answer	9 / 46 = 0.196

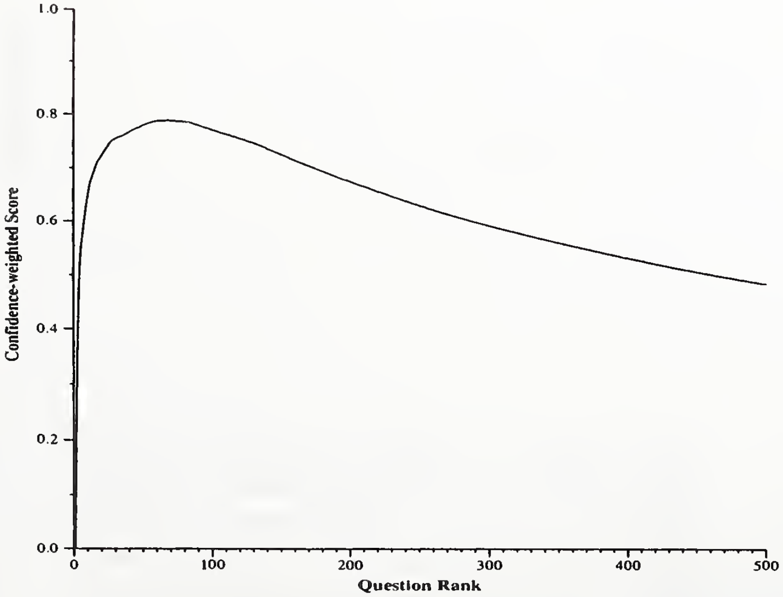


Summary Statistics	
Run ID	limsiQalir2
Num questions	500
Number wrong	336
Number unsupported	20
Number inexact	11
Number right	133
Confidence-weighted score	0.497
Precision of recognizing no answer	9 / 48 = 0.188
Recall of recognizing no answer	9 / 46 = 0.196

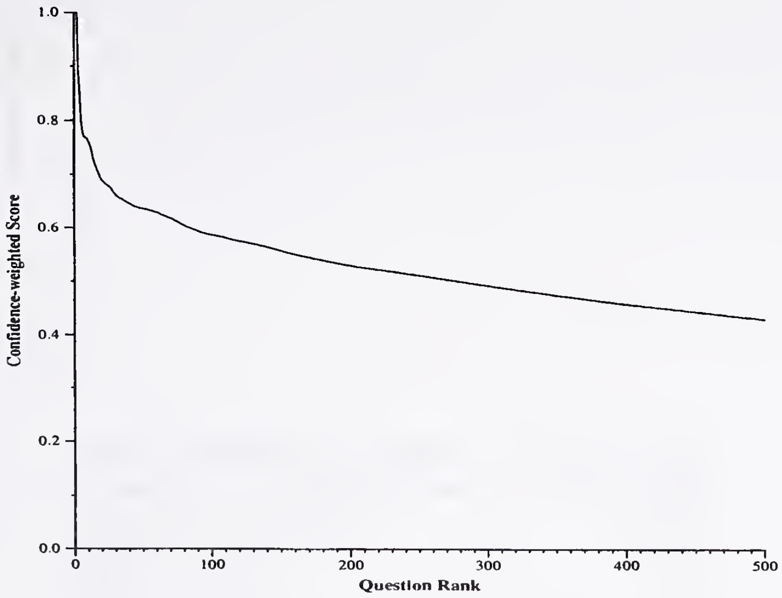


Question Answering track, Main task results — LIMSI

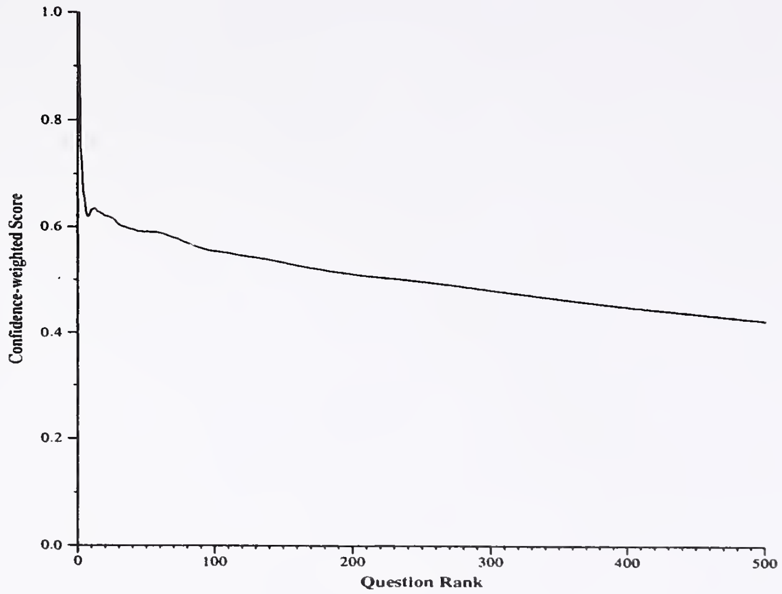
Summary Statistics	
Run ID	limsiQalir3
Num questions	500
Number wrong	330
Number unsupported	20
Number inexact	11
Number right	139
Confidence-weighted score	0.488
Precision of recognizing no answer	8 / 47 = 0.170
Recall of recognizing no answer	8 / 46 = 0.174



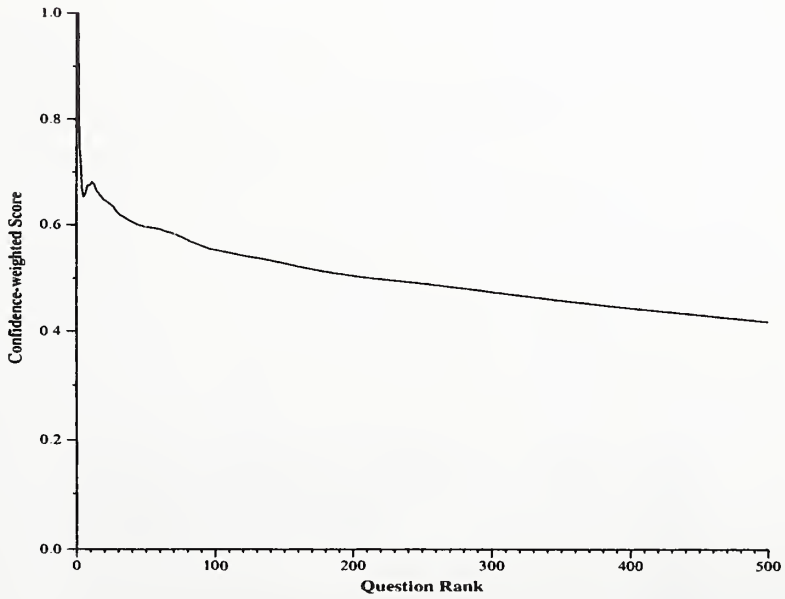
Summary Statistics	
Run ID	aranea02a
Num questions	500
Number wrong	272
Number unsupported	40
Number inexact	36
Number right	152
Confidence-weighted score	0.433
Precision of recognizing no answer	8 / 34 = 0.235
Recall of recognizing no answer	8 / 46 = 0.174



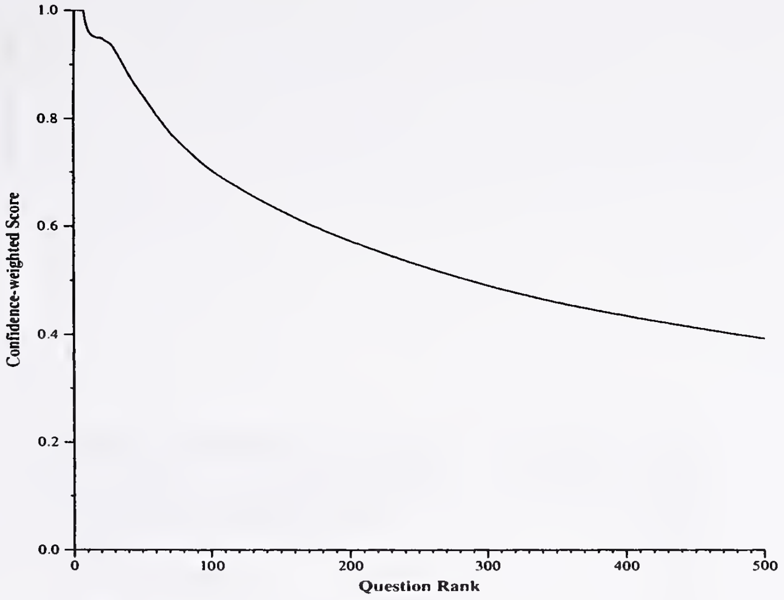
Summary Statistics	
Run ID	aranea02pbq
Num questions	500
Number wrong	272
Number unsupported	38
Number inexact	36
Number right	154
Confidence-weighted score	0.427
Precision of recognizing no answer	8 / 33 = 0.242
Recall of recognizing no answer	8 / 46 = 0.174



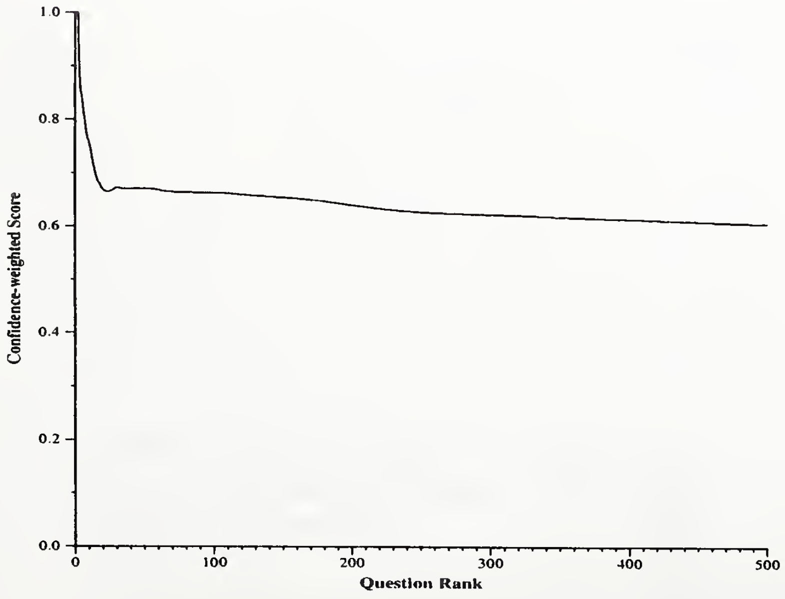
Summary Statistics	
Run ID	aranea02pc3
Num questions	500
Number wrong	273
Number unsupported	40
Number inexact	36
Number right	151
Confidence-weighted score	0.421
Precision of recognizing no answer	7 / 31 = 0.226
Recall of recognizing no answer	7 / 46 = 0.152



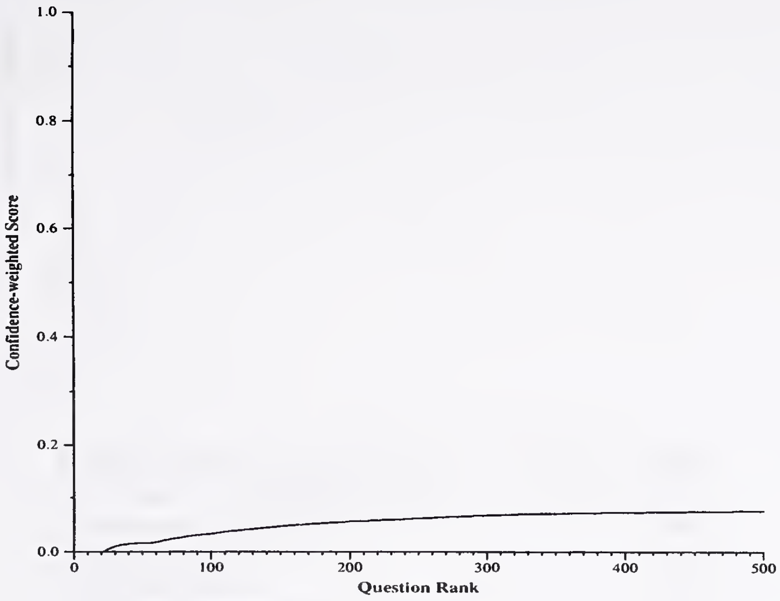
Summary Statistics	
Run ID	nuslamp2002
Num questions	500
Number wrong	351
Number unsupported	27
Number inexact	17
Number right	105
Confidence-weighted score	0.396
Precision of recognizing no answer	0 / 2 = 0.000
Recall of recognizing no answer	0 / 46 = 0.000



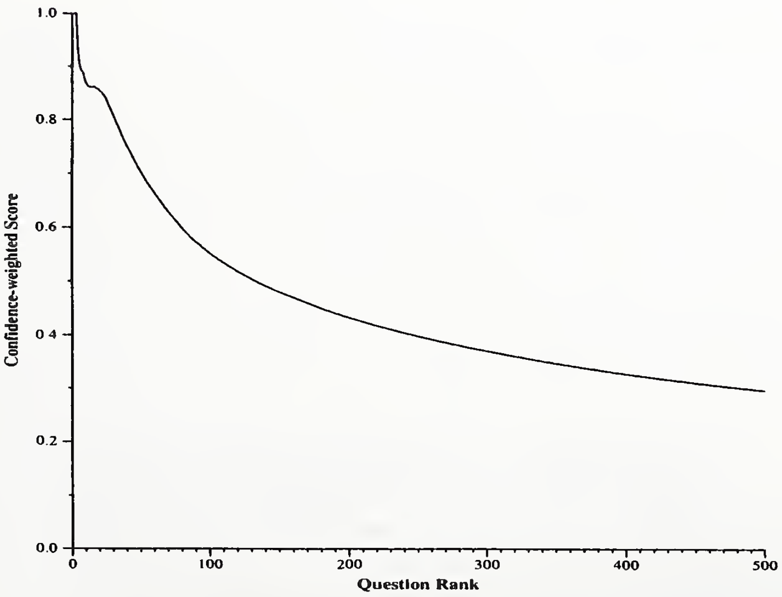
Summary Statistics	
Run ID	pris2002
Num questions	500
Number wrong	175
Number unsupported	18
Number inexact	17
Number right	290
Confidence-weighted score	0.610
Precision of recognizing no answer	41 / 170 = 0.241
Recall of recognizing no answer	41 / 46 = 0.891



Summary Statistics	
Run ID	nttcs11qam
Num questions	500
Number wrong	443
Number unsupported	4
Number inexact	6
Number right	47
Confidence-weighted score	0.076
Precision of recognizing no answer	10 / 92 = 0.109
Recall of recognizing no answer	10 / 46 = 0.217

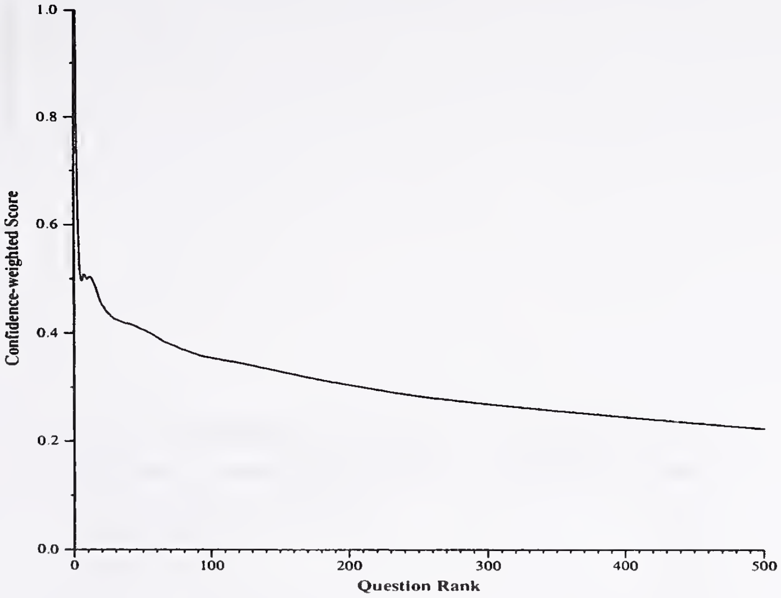


Summary Statistics	
Run ID	KLE000
Num questions	500
Number wrong	399
Number unsupported	5
Number inexact	10
Number right	86
Confidence-weighted score	0.298
Precision of recognizing no answer	15 / 93 = 0.161
Recall of recognizing no answer	15 / 46 = 0.326

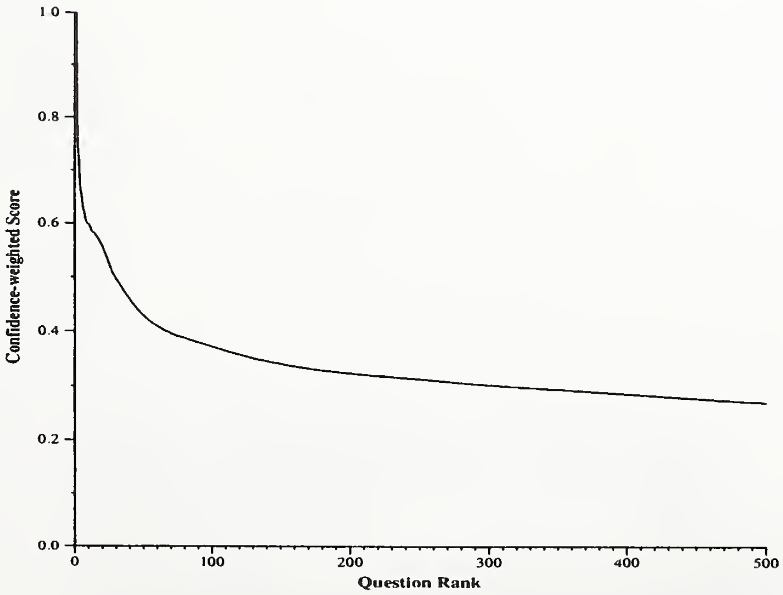


Question Answering track, Main task results — Syracuse University

Summary Statistics	
Run ID	SUT11IR1MT
Num questions	500
Number wrong	422
Number unsupported	9
Number inexact	5
Number right	64
Confidence-weighted score	0.225
Precision of recognizing no answer	12 / 72 = 0.167
Recall of recognizing no answer	12 / 46 = 0.261

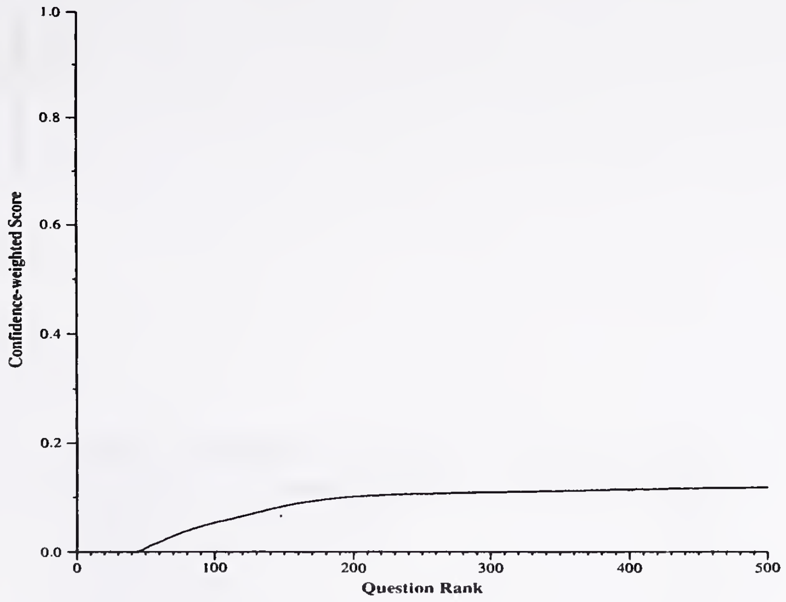


Summary Statistics	
Run ID	MITRE2002B
Num questions	500
Number wrong	377
Number unsupported	11
Number inexact	10
Number right	102
Confidence-weighted score	0.271
Precision of recognizing no answer	16 / 124 = 0.129
Recall of recognizing no answer	16 / 46 = 0.348

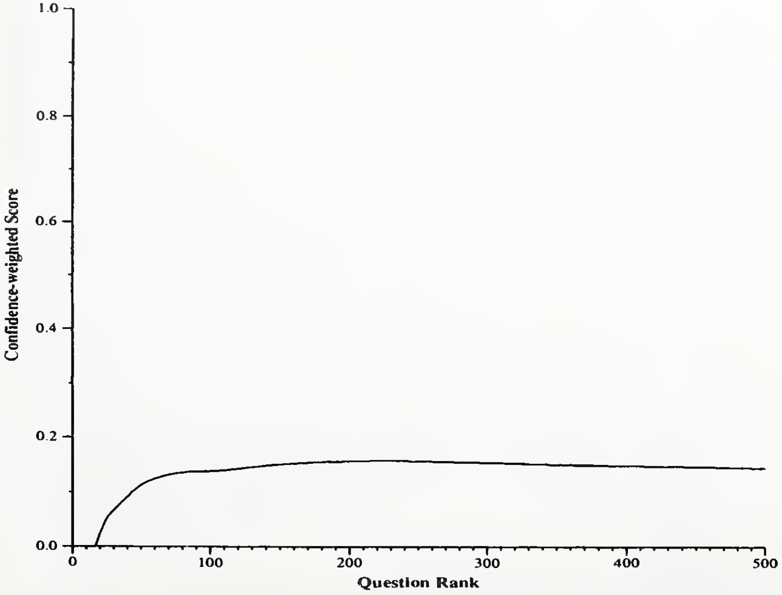


Question Answering track, Main task results — Tokyo University of Science

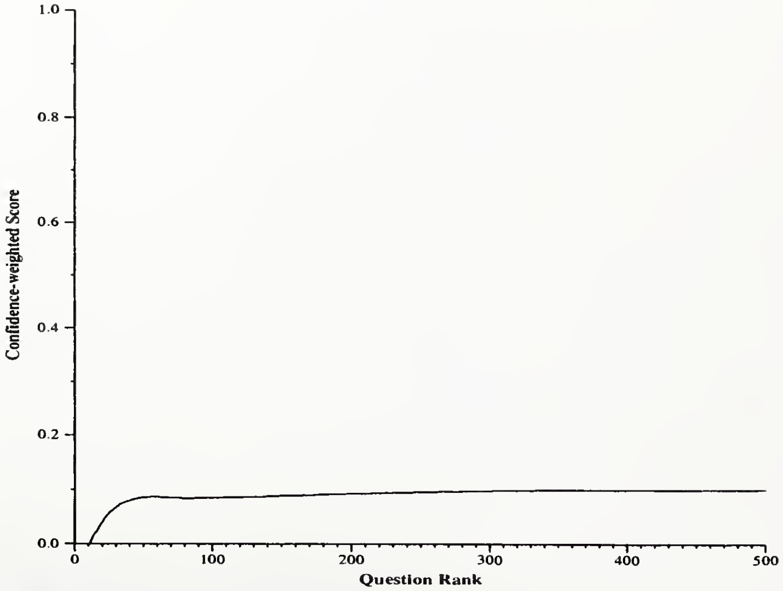
Summary Statistics	
Run ID	TUSRUN1
Num questions	500
Number wrong	405
Number unsupported	14
Number inexact	12
Number right	69
Confidence-weighted score	0.119
Precision of recognizing no answer	34 / 257 = 0.132
Recall of recognizing no answer	34 / 46 = 0.739



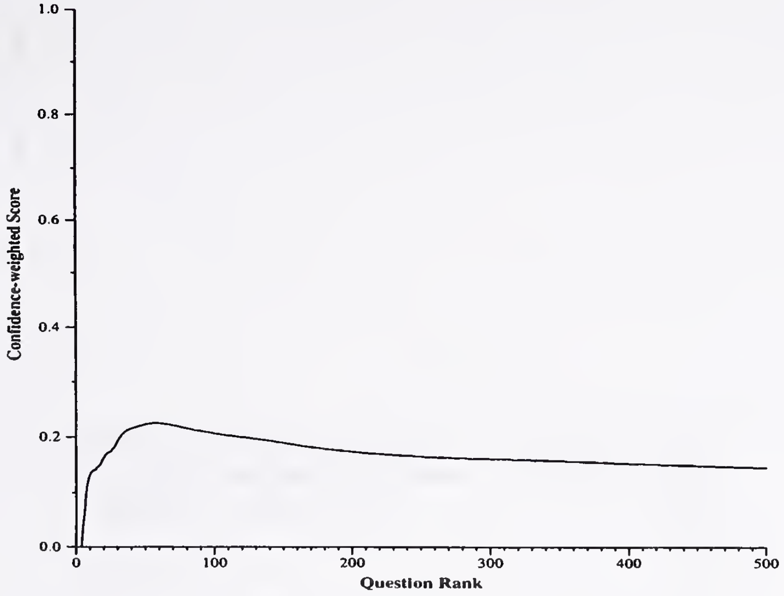
Summary Statistics	
Run ID	UAmsT11qaM1
Num questions	500
Number wrong	412
Number unsupported	21
Number inexact	3
Number right	64
Confidence-weighted score	0.145
Precision of recognizing no answer	28 / 243 = 0.115
Recall of recognizing no answer	28 / 46 = 0.609



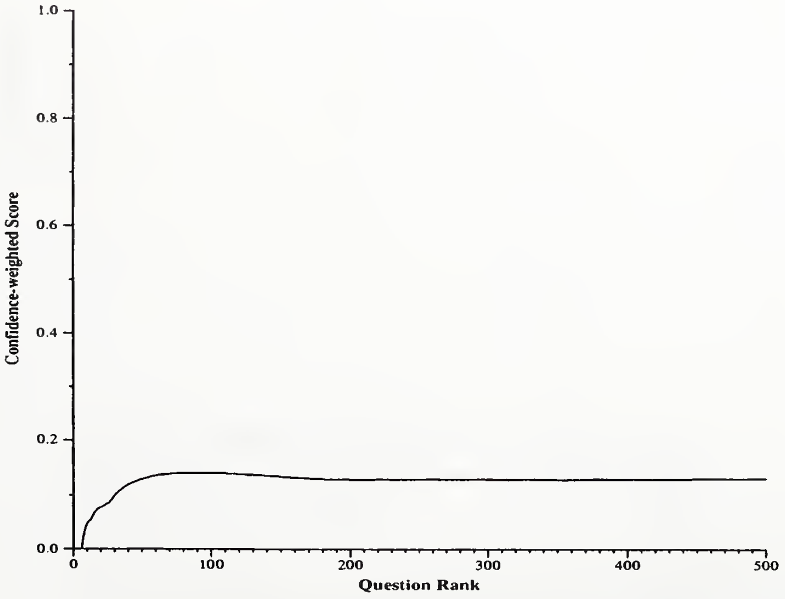
Summary Statistics	
Run ID	UAmsT11qaM2
Num questions	500
Number wrong	402
Number unsupported	32
Number inexact	10
Number right	56
Confidence-weighted score	0.101
Precision of recognizing no answer	17 / 88 = 0.193
Recall of recognizing no answer	17 / 46 = 0.370



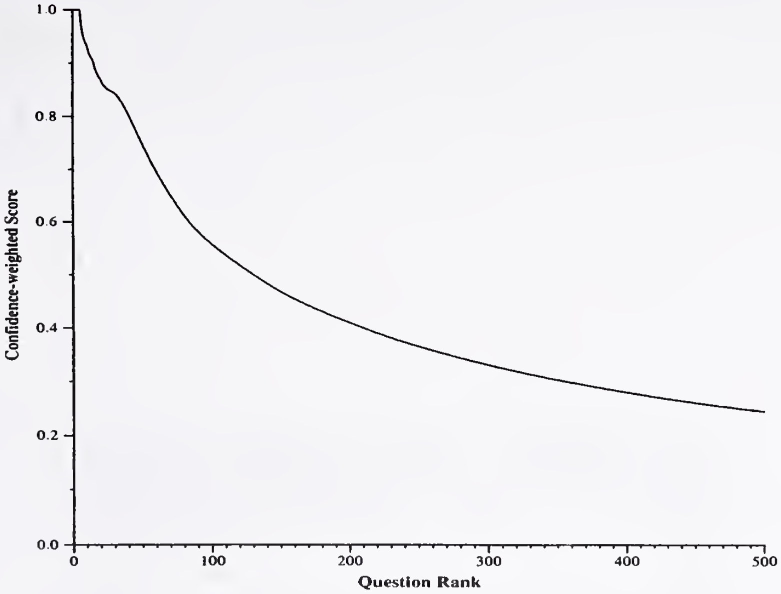
Summary Statistics	
Run ID	UAmsT11qaM3
Num questions	500
Number wrong	402
Number unsupported	32
Number inexact	10
Number right	56
Confidence-weighted score	0.146
Precision of recognizing no answer	17 / 88 = 0.193
Recall of recognizing no answer	17 / 46 = 0.370



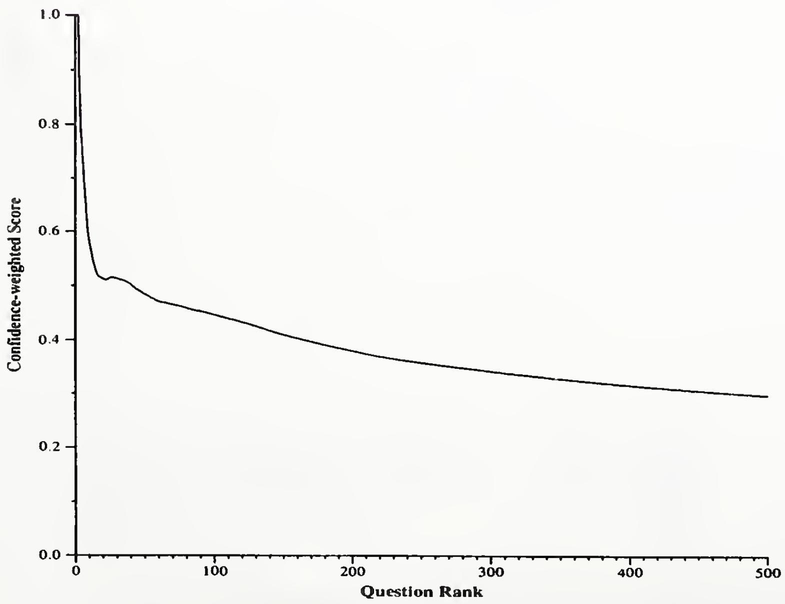
Summary Statistics	
Run ID	leria2002
Num questions	500
Number wrong	424
Number unsupported	1
Number inexact	3
Number right	72
Confidence-weighted score	0.131
Precision of recognizing no answer	46 / 450 = 0.102
Recall of recognizing no answer	46 / 46 = 1.000



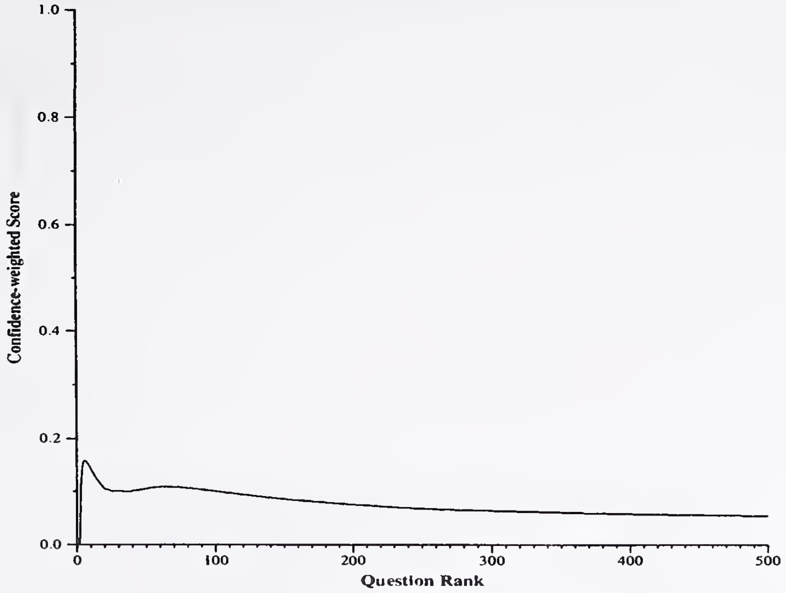
Summary Statistics	
Run ID	LIA2002a
Num questions	500
Number wrong	440
Number unsupported	4
Number inexact	4
Number right	52
Confidence-weighted score	0.246
Precision of recognizing no answer	7 / 75 = 0.093
Recall of recognizing no answer	7 / 46 = 0.152



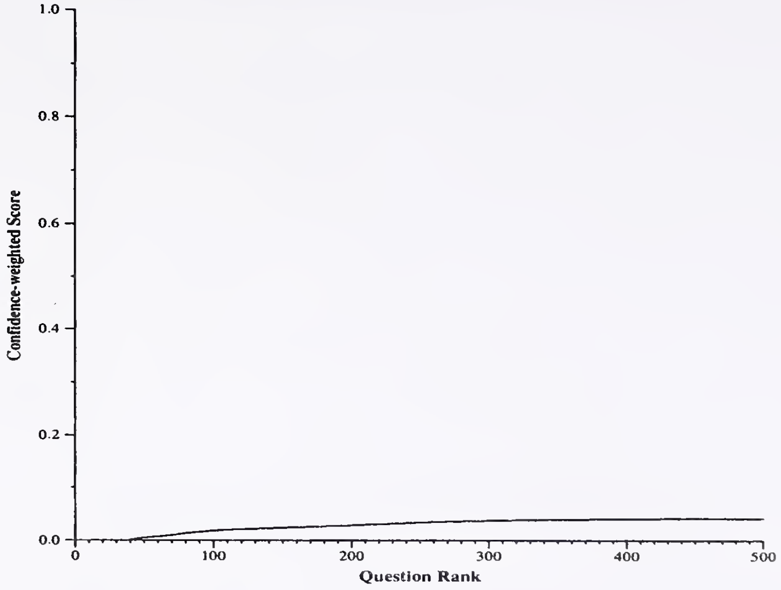
Summary Statistics	
Run ID	UIUC2002
Num questions	500
Number wrong	369
Number unsupported	13
Number inexact	9
Number right	109
Confidence-weighted score	0.299
Precision of recognizing no answer	22 / 132 = 0.167
Recall of recognizing no answer	22 / 46 = 0.478



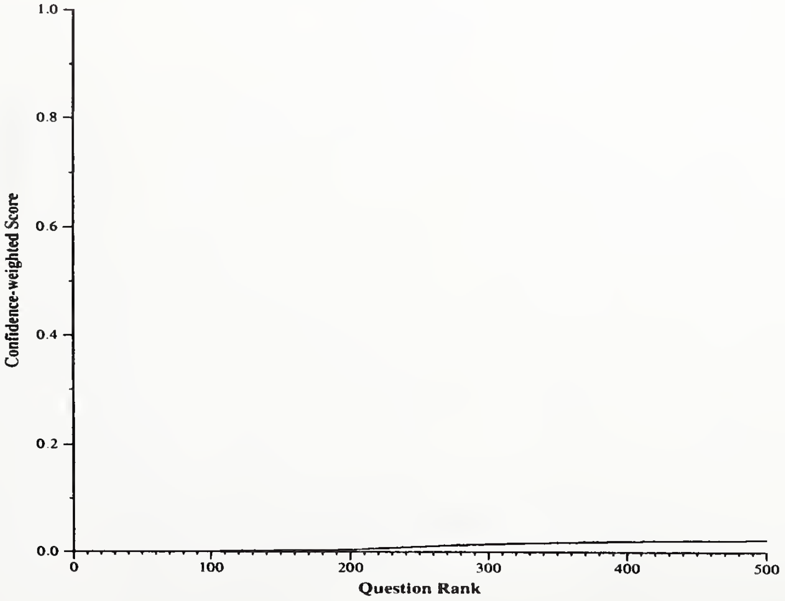
Summary Statistics	
Run ID	UIowaQA0201
Num questions	500
Number wrong	476
Number unsupported	0
Number inexact	6
Number right	18
Confidence-weighted score	0.055
Never returned a NIL response.	



Summary Statistics	
Run ID	UIowaQA0202
Num questions	500
Number wrong	474
Number unsupported	2
Number inexact	3
Number right	21
Confidence-weighted score	0.042
Never returned a NIL response.	

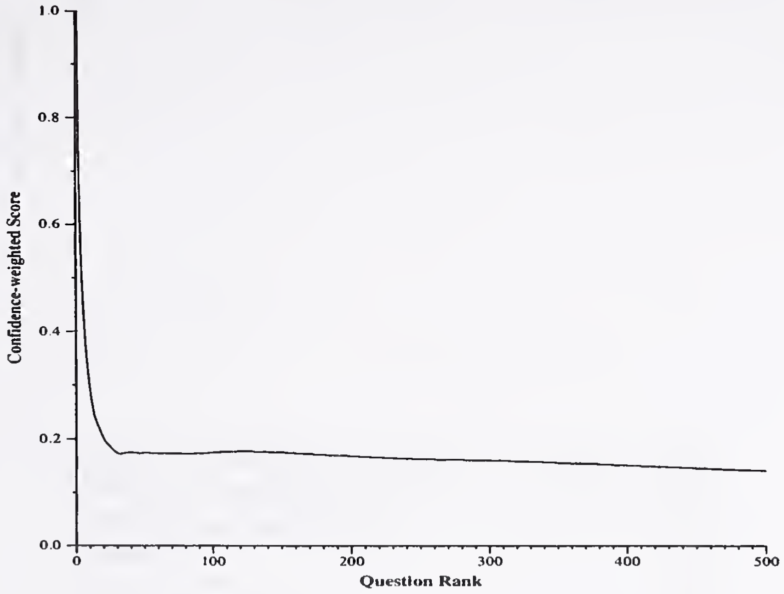


Summary Statistics	
Run ID	UIowaQA0203
Num questions	500
Number wrong	476
Number unsupported	4
Number inexact	5
Number right	15
Confidence-weighted score	0.023
Never returned a NIL response.	

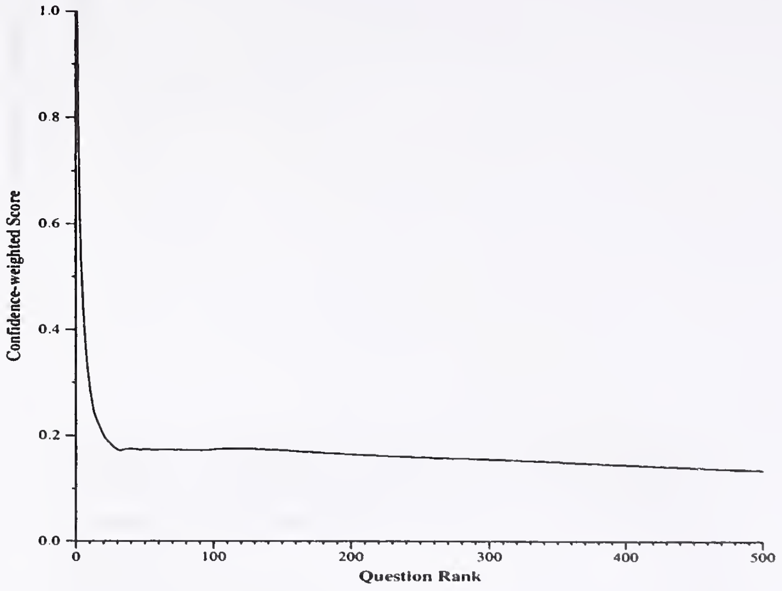


Question Answering track, Main task results — University of Limerick

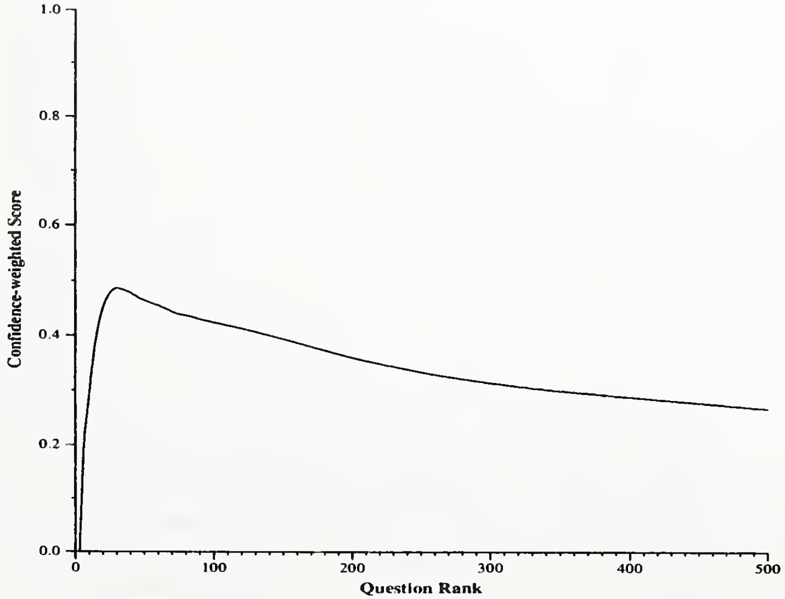
Summary Statistics	
Run ID	DLT02QA01
Num questions	500
Number wrong	439
Number unsupported	6
Number inexact	5
Number right	50
Confidence-weighted score	0.142
Precision of recognizing no answer	3 / 9 = 0.333
Recall of recognizing no answer	3 / 46 = 0.065



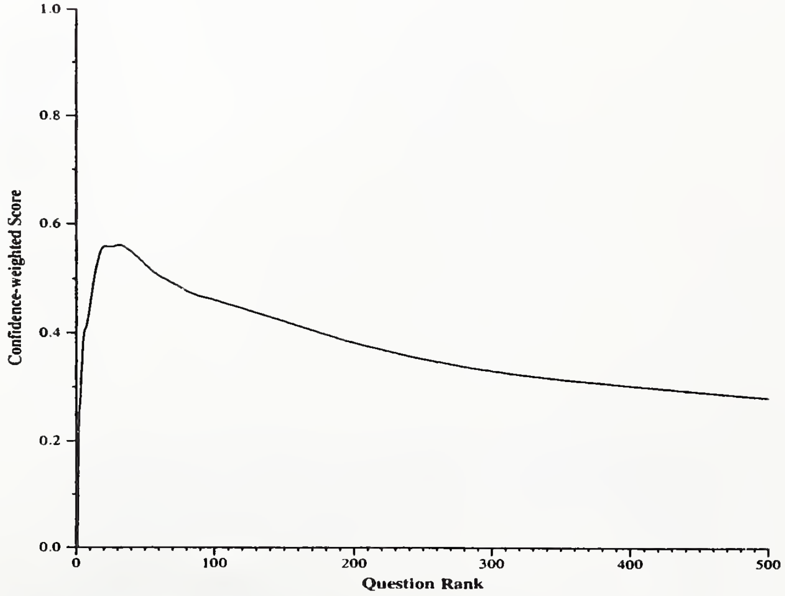
Summary Statistics	
Run ID	DLT02QA02
Num questions	500
Number wrong	441
Number unsupported	8
Number inexact	4
Number right	47
Confidence-weighted score	0.136
Precision of recognizing no answer	3 / 9 = 0.333
Recall of recognizing no answer	3 / 46 = 0.065



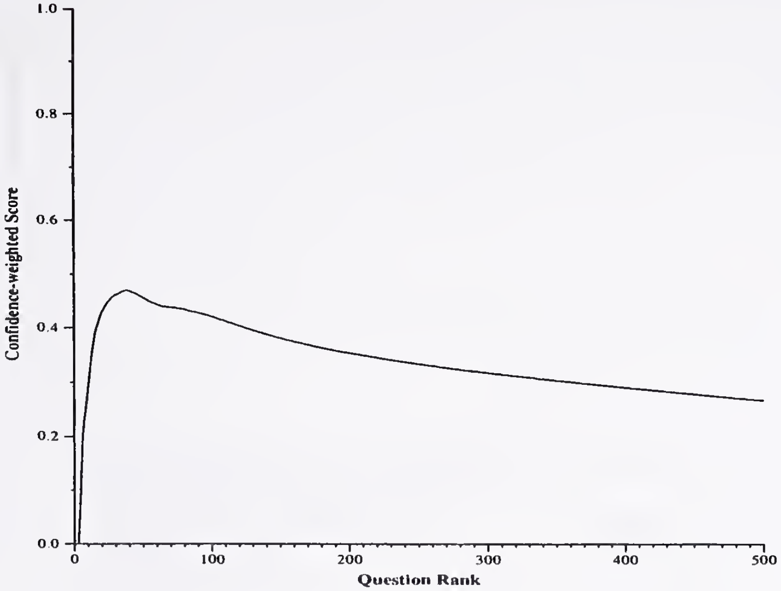
Summary Statistics	
Run ID	NSIRG
Num questions	500
Number wrong	388
Number unsupported	16
Number inexact	12
Number right	84
Confidence-weighted score	0.268
Precision of recognizing no answer	25 / 195 = 0.128
Recall of recognizing no answer	25 / 46 = 0.543



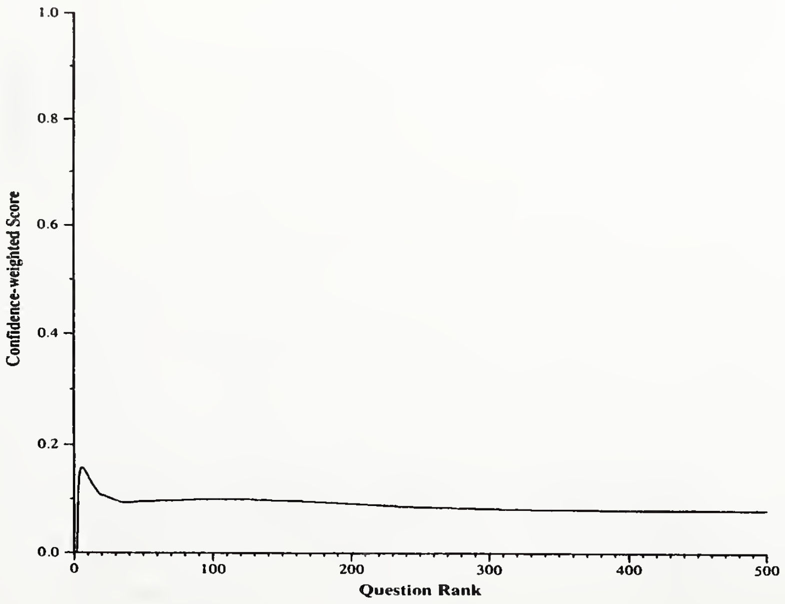
Summary Statistics	
Run ID	NSIRGN
Num questions	500
Number wrong	379
Number unsupported	18
Number inexact	14
Number right	89
Confidence-weighted score	0.283
Precision of recognizing no answer	23 / 175 = 0.131
Recall of recognizing no answer	23 / 46 = 0.500



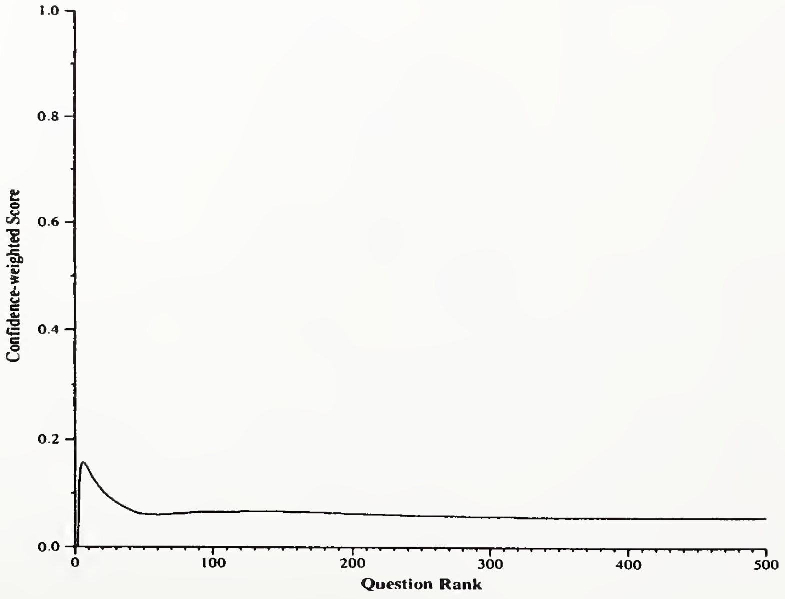
Summary Statistics	
Run ID	NSIRN
Num questions	500
Number wrong	404
Number unsupported	7
Number inexact	11
Number right	78
Confidence-weighted score	0.269
Precision of recognizing no answer	12 / 63 = 0.190
Recall of recognizing no answer	12 / 46 = 0.261



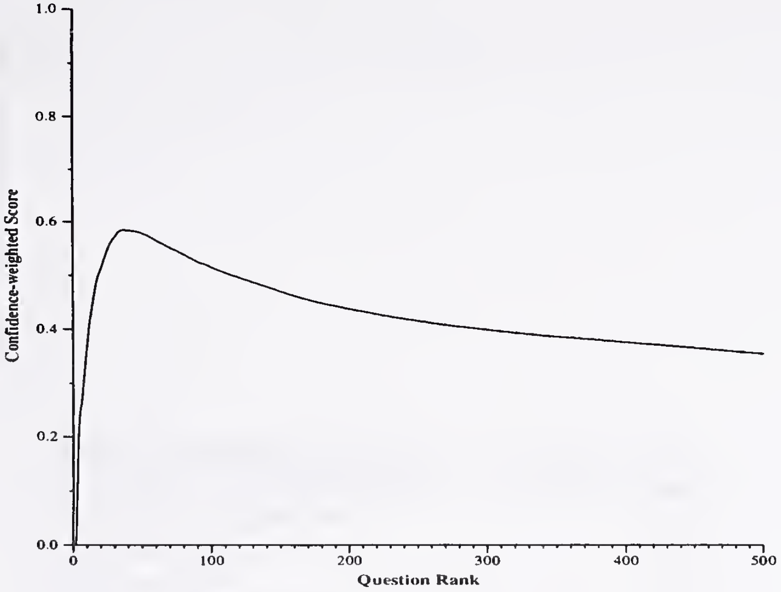
Summary Statistics	
Run ID	UdeMmainNoW
Num questions	500
Number wrong	454
Number unsupported	2
Number inexact	7
Number right	37
Confidence-weighted score	0.080
Never returned a NIL response.	



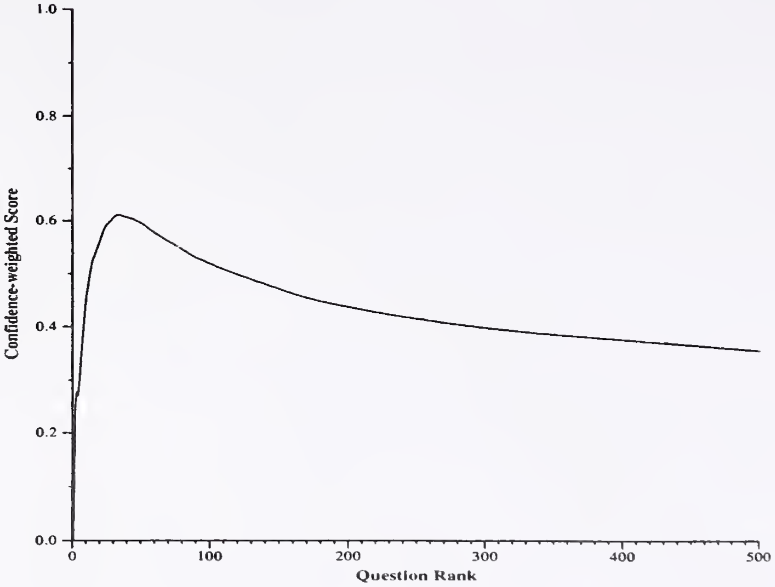
Summary Statistics	
Run ID	UdeMmainWeb
Num questions	500
Number wrong	425
Number unsupported	40
Number inexact	6
Number right	29
Confidence-weighted score	0.057
Never returned a NIL response.	



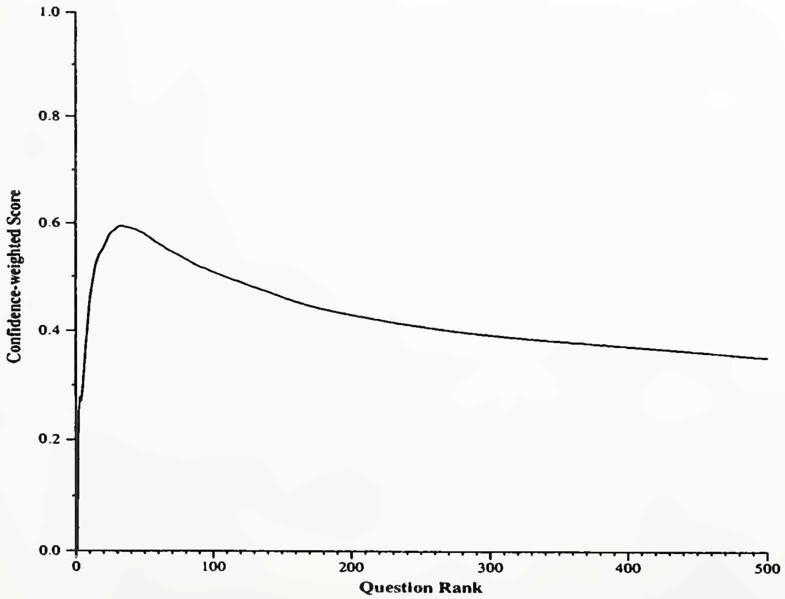
Summary Statistics	
Run ID	pqas21
Num questions	500
Number wrong	355
Number unsupported	4
Number inexact	11
Number right	130
Confidence-weighted score	0.357
Precision of recognizing no answer	31 / 209 = 0.148
Recall of recognizing no answer	31 / 46 = 0.674



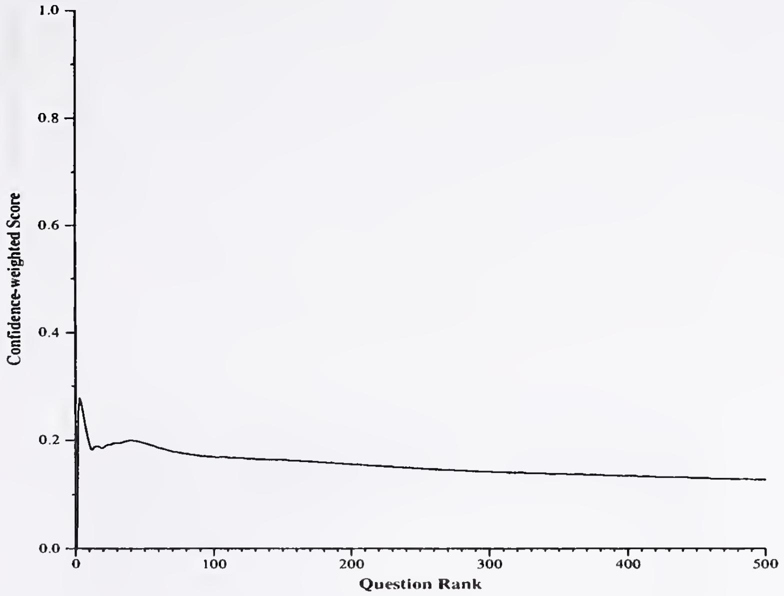
Summary Statistics	
Run ID	pqas22
Num questions	500
Number wrong	352
Number unsupported	4
Number inexact	11
Number right	133
Confidence-weighted score	0.358
Precision of recognizing no answer	31 / 214 = 0.145
Recall of recognizing no answer	31 / 46 = 0.674



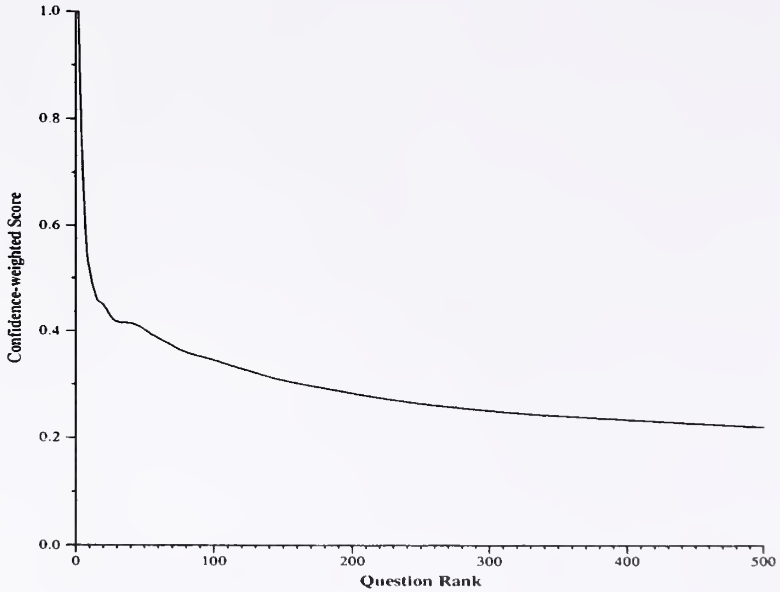
Summary Statistics	
Run ID	pqas23
Num questions	500
Number wrong	353
Number unsupported	4
Number inexact	12
Number right	131
Confidence-weighted score	0.354
Precision of recognizing no answer	31 / 213 = 0.146
Recall of recognizing no answer	31 / 46 = 0.674



Summary Statistics	
Run ID	sheft11mo3
Num questions	500
Number wrong	422
Number unsupported	9
Number inexact	18
Number right	51
Confidence-weighted score	0.128
Precision of recognizing no answer	6 / 37 = 0.162
Recall of recognizing no answer	6 / 46 = 0.130

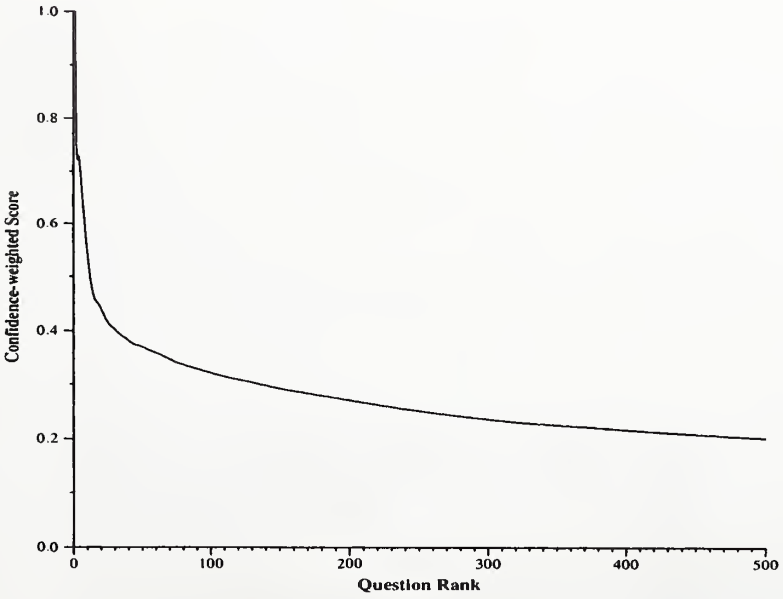


Summary Statistics	
Run ID	sheft11mog1
Num questions	500
Number wrong	389
Number unsupported	11
Number inexact	20
Number right	80
Confidence-weighted score	0.222
Precision of recognizing no answer	3 / 20 = 0.150
Recall of recognizing no answer	3 / 46 = 0.065

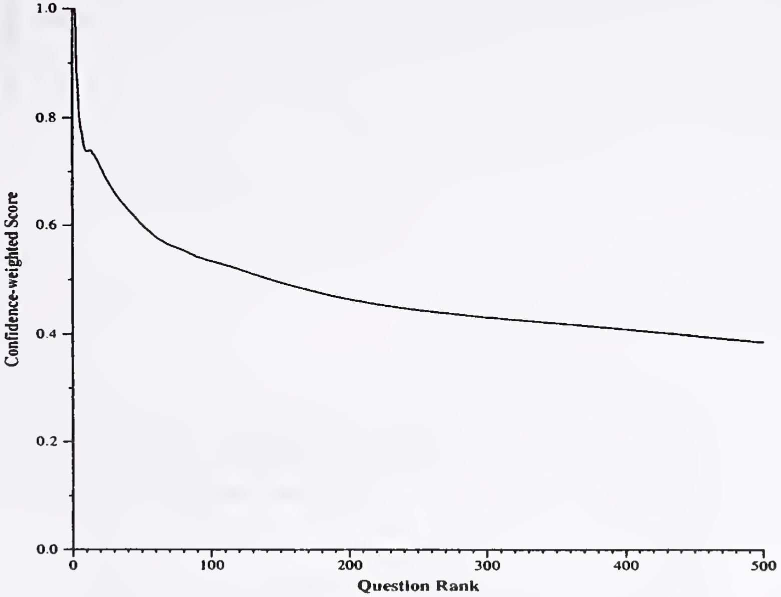


Question Answering track, Main task results — University of Sheffield, Western Bank

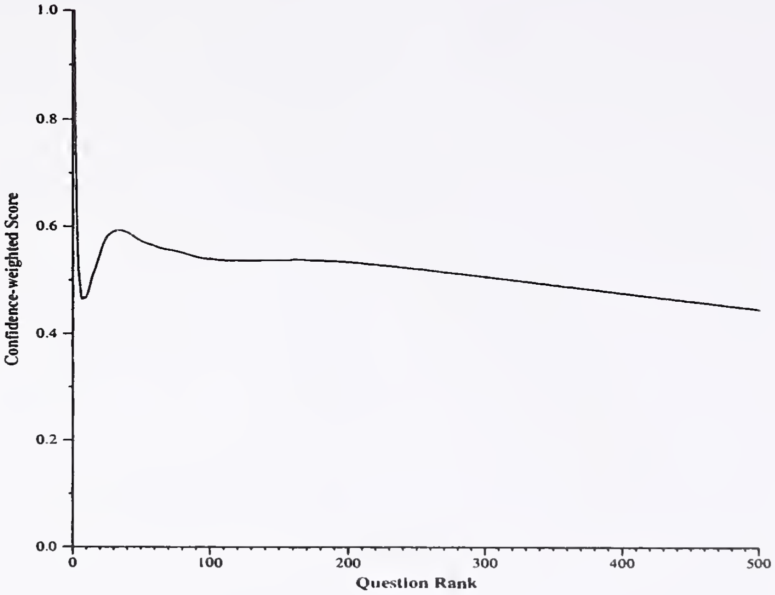
Summary Statistics	
Run ID	sheft11mog3
Num questions	500
Number wrong	394
Number unsupported	12
Number inexact	22
Number right	72
Confidence-weighted score	0.203
Precision of recognizing no answer	6 / 40 = 0.150
Recall of recognizing no answer	6 / 46 = 0.130



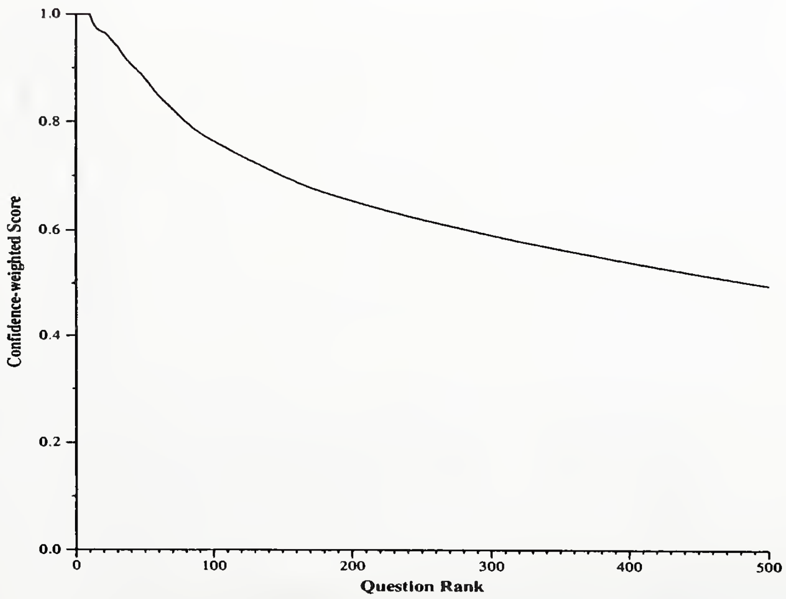
Summary Statistics	
Run ID	ilv02t
Num questions	500
Number wrong	334
Number unsupported	13
Number inexact	14
Number right	139
Confidence-weighted score	0.389
Precision of recognizing no answer	2 / 2 = 1.000
Recall of recognizing no answer	2 / 46 = 0.043



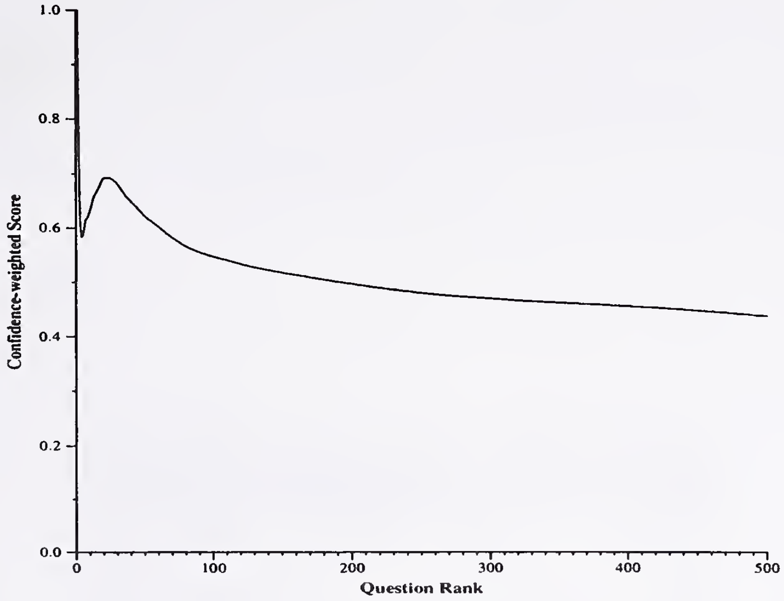
Summary Statistics	
Run ID	ilv02wt
Num questions	500
Number wrong	290
Number unsupported	38
Number inexact	18
Number right	154
Confidence-weighted score	0.450
Precision of recognizing no answer	2 / 2 = 1.000
Recall of recognizing no answer	2 / 46 = 0.043



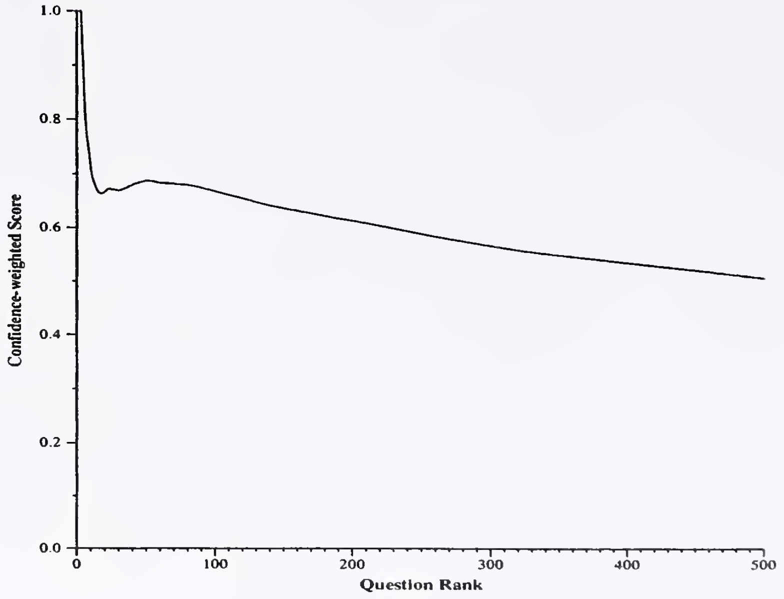
Summary Statistics	
Run ID	isi02
Num questions	500
Number wrong	302
Number unsupported	34
Number inexact	15
Number right	149
Confidence-weighted score	0.498
Precision of recognizing no answer	5 / 13 = 0.385
Recall of recognizing no answer	5 / 46 = 0.109



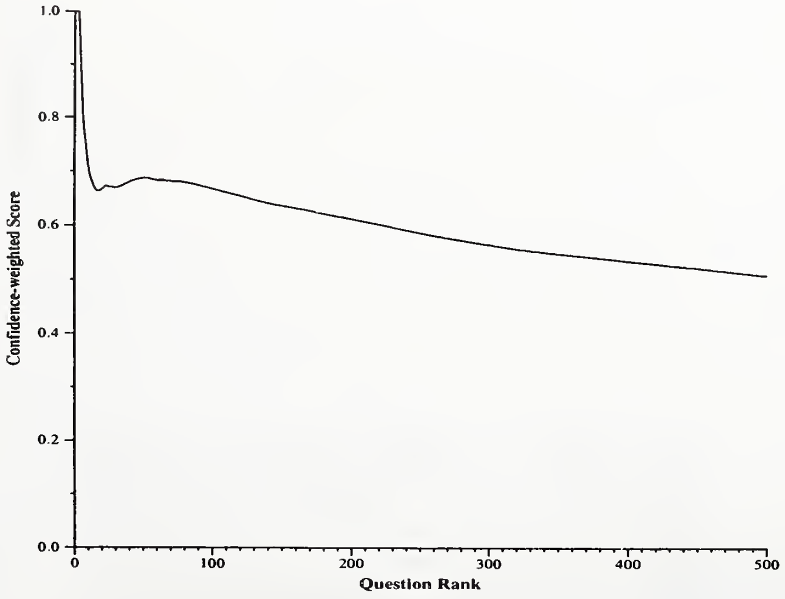
Summary Statistics	
Run ID	uwmtB1
Num questions	500
Number wrong	290
Number unsupported	17
Number inexact	26
Number right	167
Confidence-weighted score	0.441
Precision of recognizing no answer	0 / 12 = 0.000
Recall of recognizing no answer	0 / 46 = 0.000



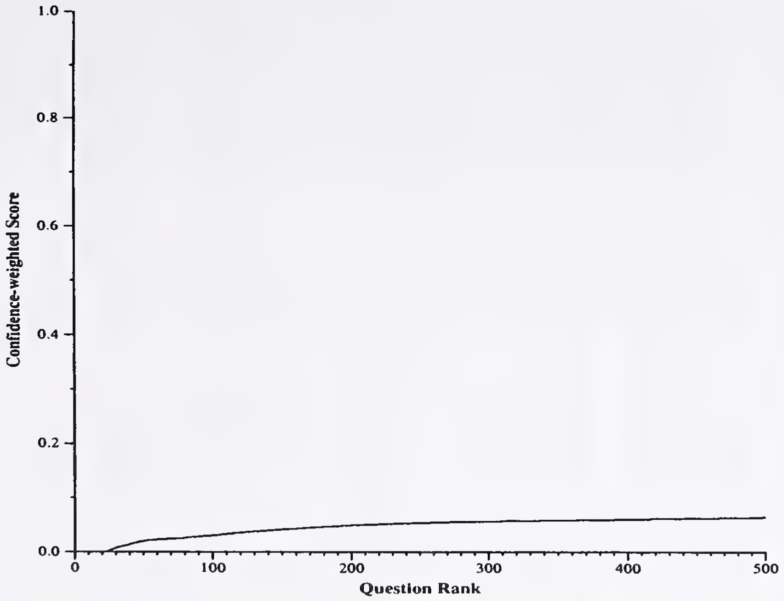
Summary Statistics	
Run ID	uwmtB2
Num questions	500
Number wrong	277
Number unsupported	22
Number inexact	18
Number right	183
Confidence-weighted score	0.511
Precision of recognizing no answer	0 / 12 = 0.000
Recall of recogiizing no answer	0 / 46 = 0.000



Summary Statistics	
Run ID	uwmtB3
Num questions	500
Number wrong	276
Number unsupported	20
Number inexact	20
Number right	184
Confidence-weighted score	0.512
Precision of recognizing no answer	0 / 12 = 0.000
Recall of recognizing no answer	0 / 46 = 0.000



Summary Statistics	
Run ID	yorkqa01
Num questions	500
Number wrong	437
Number unsupported	6
Number inexact	15
Number right	42
Confidence-weighted score	0.065
Never returned a NIL response.	



TREC 2002 Video Track Runs

Shot boundary determination

545068 total frames   2090 total shot boundaries   70% cuts

	All		Cuts		Gradual				
Sys	Rec	Prec	Rec	Prec	Rec	Prec	F-Rec	F-Prec	
CLIPS-0	0.941	0.312	0.941	0.445	0.942	0.177	0.752	0.639	CLIPS-Imag Lab
CLIPS-1	0.929	0.485	0.934	0.591	0.915	0.332	0.720	0.758	
CLIPS-2	0.915	0.658	0.926	0.733	0.888	0.517	0.698	0.851	
CLIPS-3	0.898	0.780	0.915	0.826	0.853	0.677	0.670	0.880	
CLIPS-4	0.874	0.848	0.900	0.895	0.809	0.739	0.634	0.897	
CLIPS-5	0.842	0.888	0.876	0.935	0.756	0.776	0.623	0.903	
CLIPS-6	0.804	0.906	0.839	0.952	0.717	0.793	0.598	0.914	
CLIPS-7	0.739	0.915	0.773	0.960	0.653	0.801	0.577	0.923	
CLIPS-8	0.653	0.919	0.675	0.964	0.597	0.811	0.574	0.930	
CLIPS-9	0.542	0.917	0.542	0.964	0.541	0.814	0.543	0.941	
Fudan_SB_01	0.823	0.885	0.887	0.927	0.662	0.767	0.653	0.696	Fudan University
Fudan_SB_02	0.816	0.872	0.885	0.917	0.640	0.744	0.672	0.630	
Fudan_SB_03	0.818	0.887	0.894	0.926	0.626	0.771	0.603	0.767	
Fudan_SB_04	0.841	0.871	0.915	0.905	0.651	0.768	0.651	0.696	
Fudan_SB_05	0.821	0.885	0.884	0.928	0.662	0.764	0.653	0.696	
Fudan_SB_06	0.831	0.882	0.887	0.933	0.689	0.748	0.713	0.594	
Fudan_SB_07	0.823	0.884	0.887	0.926	0.662	0.765	0.654	0.696	
Fudan_SB_08	0.831	0.882	0.887	0.933	0.689	0.748	0.713	0.594	
Fudan_SB_09	0.789	0.874	0.859	0.926	0.613	0.730	0.712	0.602	
Fudan_SB_10	0.781	0.860	0.860	0.894	0.582	0.754	0.540	0.783	
IBM_VID02_Alm1	0.857	0.773	0.925	0.795	0.685	0.706	0.481	0.941	IBM Research
IBM_VID02_NO33	0.870	0.837	0.925	0.868	0.729	0.751	0.514	0.935	
IBM_VID02_NO47	0.884	0.828	0.934	0.871	0.758	0.718	0.567	0.887	
IBM_VID02_NO48	0.729	0.866	0.930	0.870	0.220	0.828	0.443	0.969	
IBM_VID02_NO58	0.877	0.821	0.950	0.844	0.690	0.751	0.509	0.852	
ICMKM-01	0.826	0.843	0.883	0.895	0.682	0.707	0.673	0.608	Imperial College
ICMKM-02	0.845	0.798	0.889	0.863	0.733	0.648	0.658	0.618	
ICMKM-03	0.859	0.720	0.893	0.803	0.773	0.553	0.650	0.612	
ICMKM-04	0.825	0.813	0.888	0.880	0.665	0.645	0.471	0.603	
ICMKM-05	0.833	0.755	0.891	0.832	0.685	0.578	0.477	0.444	
ICMKM-06	0.836	0.688	0.885	0.755	0.711	0.536	0.477	0.356	
MSSD01	0.749	0.753	0.879	0.763	0.420	0.707	0.532	0.918	Microsoft Research Asia
MSSD02	0.754	0.709	0.878	0.735	0.440	0.599	0.528	0.921	
MSSD03	0.760	0.700	0.884	0.725	0.447	0.596	0.515	0.922	
MSSD05	0.748	0.753	0.879	0.763	0.418	0.706	0.531	0.918	
MSSD06	0.739	0.673	0.879	0.667	0.382	0.713	0.568	0.919	
MSSD07	0.685	0.821	0.883	0.832	0.181	0.709	0.456	0.892	
MSSD08	0.725	0.768	0.841	0.772	0.433	0.751	0.531	0.922	
MSSD09	0.178	0.373	0.180	0.460	0.173	0.249	0.529	0.034	
MSSD10	0.723	0.805	0.813	0.806	0.497	0.799	0.655	0.604	
nus1	0.621	0.615	0.742	0.670	0.313	0.411	0.301	0.833	National Univ. of Singapore
nus2	0.594	0.614	0.707	0.693	0.306	0.369	0.331	0.848	
rmit_1	0.620	0.928	0.865	0.928	0.000	0.000	0.000	0.000	RMIT University
rmit_2	0.806	0.784	0.842	0.914	0.716	0.551	0.656	0.702	
rmit_3	0.648	0.828	0.904	0.828	0.000	0.000	0.000	0.000	
rmit_4	0.594	0.958	0.828	0.959	0.002	0.333	0.786	0.611	
rmit_5	0.748	0.830	0.799	0.936	0.618	0.604	0.692	0.718	
rmit_6	0.641	0.834	0.891	0.834	0.007	0.800	0.673	0.529	
rmit_7	0.725	0.792	0.795	0.925	0.547	0.517	0.676	0.729	
rmit_8	0.707	0.813	0.784	0.937	0.511	0.537	0.687	0.715	
rmit_9	0.687	0.871	0.954	0.877	0.008	0.294	0.506	0.854	
rmit_10	0.655	0.940	0.913	0.942	0.000	0.000	0.000	0.000	
TZI-UBremen	0.592	0.708	0.793	0.734	0.083	0.380	0.247	0.765	TZI-University of Bremen

TREC 2002 Video Track Runs (cont.)

Feature extraction

ID	Priority	Affiliation
CMU.r1	1	Carnegie Mellon Univ.
CMU.r2	2	
CMU.r3	3	
CLIPS-LIT-GEOD	1	CLIPS_IMAG Lab
CLIPS-LIT-LIMSU	2	
DCUFE2002	1	Dublin City Univ.
Eurecom1	1	Eurecom
Fudan_FE_Sys1	1	Fudan University
Fudan_FE_Sys2	2	
IBM-1	1	IBM Research
IBM-2	2	
MediaMill1	1	
MediaMill2	2	
MSRA	1	Microsoft Research Asia
TZI_univ_bremen	1	Univ. of Bremen
UMD1	1	Univ. of Maryland
UnivO_MT1	1	Univ. of Oulu
UnivO_MT2	2	

Feature definitions

If the feature is true for some frame (sequence) within the shot, then it is true for the shot.

1. **Outdoors**: Segment contains a recognizably outdoor location, i.e., one outside of buildings. Should exclude all scenes that are indoors or are close-ups of objects (even if the objects are outdoor).
2. **Indoors**: Segment contains a recognizably indoor location, i.e., inside a building. Should exclude all scenes that are outdoors or are close-ups of objects (even if the objects are indoor).
3. **Face**: Segment contains at least one human face with the nose, mouth, and both eyes visible. Pictures of a face meeting the above conditions count.
4. **People**: Segment contains a group of two more humans, each of which is at least partially visible and is recognizable as a human.
5. **Cityscape**: Segment contains a recognizably city/urban/suburban setting.
6. **Landscape**: Segment contains a predominantly natural inland setting, i.e., one with little or no evidence of development by humans. For example, scenes consisting mostly of plowed/planted fields, pastures, orchards would be excluded. Some buildings, if small features on the overall landscape, should be OK. Scenes with bodies of water that are clearly inland may be included.
7. **Text Overlay**: Segment contains superimposed text large enough to be read.
8. **Speech**: A human voice uttering words is recognizable as such in this segment
9. **Instrumental Sound**: Sound produced by one or more musical instruments is recognizable as such in this segment. Included are percussion instruments.
10. **Monologue**: Segment contains an event in which a single person is at least partially visible and speaks for a long time without interruption by another speaker. Pauses are ok if short.

TREC 2002 Video Track
Feature Extraction Results

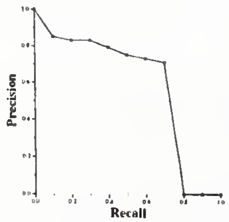
Run ID:
  CMU.r1

Processing type:
  automatic

System training type:
  B (no knowledge of test set)

Feature: 1

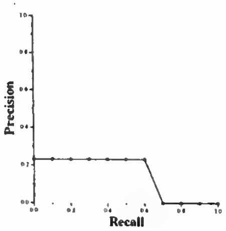
Total shots returned:	1000
Total shots with feature:	962
Total shots with feature returned:	698
Precision at shots with feature:	0.7037
Average precision:	0.5734



Interpolated recall	precision	Precision at n shots	
0.0	1.0000	5	1.0000
0.1	0.8511	10	1.0000
0.2	0.8305	15	0.9333
0.3	0.8305	20	0.9000
0.4	0.7910	30	0.8667
0.5	0.7492	100	0.8200
0.6	0.7292	200	0.8450
0.7	0.7072	500	0.7820
0.8	0.0000	1000	0.6980
0.9	0.0000		
1.0	0.0000		

Feature: 2

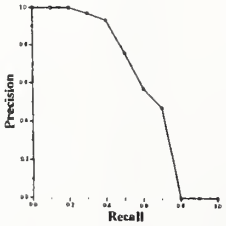
Total shots returned:	983
Total shots with feature:	351
Total shots with feature returned:	226
Precision at shots with feature:	0.1738
Average precision:	0.1278



Interpolated recall	precision	Precision at n shots	
0.0	0.2312	5	0.0000
0.1	0.2312	10	0.0000
0.2	0.2312	15	0.0667
0.3	0.2312	20	0.0500
0.4	0.2312	30	0.0667
0.5	0.2312	100	0.1400
0.6	0.2312	200	0.1800
0.7	0.0000	500	0.2100
0.8	0.0000	1000	0.2260
0.9	0.0000		
1.0	0.0000		

Feature: 3

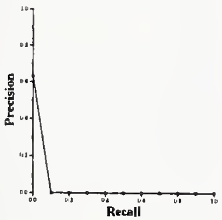
Total shots returned:	689
Total shots with feature:	415
Total shots with feature returned:	303
Precision at shots with feature:	0.5928
Average precision:	0.6133



Interpolated recall	precision	Precision at n shots	
0.0	1.0000	5	1.0000
0.1	1.0000	10	1.0000
0.2	1.0000	15	1.0000
0.3	0.9708	20	1.0000
0.4	0.9341	30	1.0000
0.5	0.7609	100	0.9900
0.6	0.5737	200	0.8950
0.7	0.4696	500	0.5300
0.8	0.0000	1000	0.3030
0.9	0.0000		
1.0	0.0000		

Feature: 4

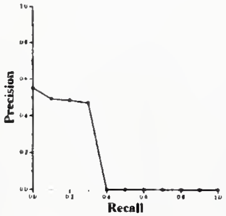
Total shots returned:	105
Total shots with feature:	486
Total shots with feature returned:	48
Precision at shots with feature:	0.0988
Average precision:	0.0496



Interpolated recall	precision	Precision at n shots	
0.0	0.6316	5	0.4000
0.1	0.0000	10	0.5000
0.2	0.0000	15	0.5333
0.3	0.0000	20	0.6000
0.4	0.0000	30	0.4667
0.5	0.0000	100	0.4500
0.6	0.0000	200	0.2400
0.7	0.0000	500	0.0960
0.8	0.0000	1000	0.0480
0.9	0.0000		
1.0	0.0000		

Feature: 5

Total shots returned:	381
Total shots with feature:	521
Total shots with feature returned:	168
Precision at shots with feature:	0.3225
Average precision:	0.1538

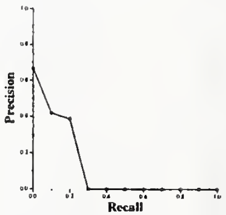


Interpolated recall	precision	Precision at n shots	
0.0	0.5517	5	0.2000
0.1	0.4925	10	0.3000
0.2	0.4846	15	0.4000
0.3	0.4706	20	0.5000
0.4	0.0000	30	0.4667
0.5	0.0000	100	0.4800
0.6	0.0000	200	0.4900
0.7	0.0000	500	0.3360
0.8	0.0000	1000	0.1680
0.9	0.0000		
1.0	0.0000		

No submission for feature 6

Feature: 7

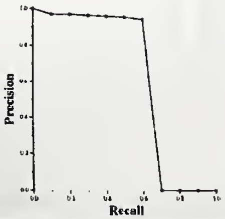
Total shots returned:	66
Total shots with feature:	110
Total shots with feature returned:	25
Precision at shots with feature:	0.2273
Average precision:	0.0918



Interpolated recall	precision	Precision at n shots	
0.0	0.6667	5	0.6000
0.1	0.4186	10	0.4000
0.2	0.3871	15	0.3333
0.3	0.0000	20	0.3000
0.4	0.0000	30	0.4000
0.5	0.0000	100	0.2500
0.6	0.0000	200	0.1250
0.7	0.0000	500	0.0500
0.8	0.0000	1000	0.0250
0.9	0.0000		
1.0	0.0000		

Feature: 8

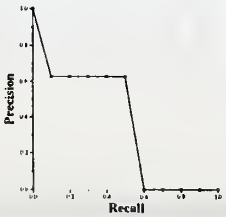
Total shots returned:	1000
Total shots with feature:	1382
Total shots with feature returned:	936
Precision at shots with feature:	0.6773
Average precision:	0.6420



Interpolated recall	precision	Precision at n shots	
0.0	1.0000	5	1.0000
0.1	0.9697	10	0.9000
0.2	0.9697	15	0.8000
0.3	0.9640	20	0.8500
0.4	0.9602	30	0.9000
0.5	0.9555	100	0.9200
0.6	0.9424	200	0.9550
0.7	0.0000	500	0.9600
0.8	0.0000	1000	0.9360
0.9	0.0000		
1.0	0.0000		

Feature: 9

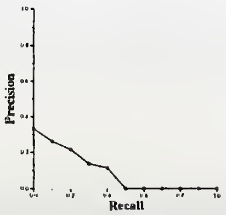
Total shots returned:	1000
Total shots with feature:	1221
Total shots with feature returned:	625
Precision at shots with feature:	0.5119
Average precision:	0.2571



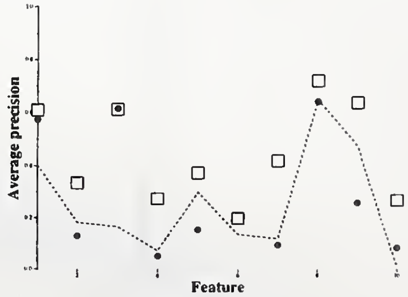
Interpolated recall	precision	Precision at n shots	
0.0	1.0000	5	0.6000
0.1	0.6271	10	0.5000
0.2	0.6271	15	0.3333
0.3	0.6271	20	0.3000
0.4	0.6271	30	0.3000
0.5	0.6271	100	0.2000
0.6	0.0000	200	0.1950
0.7	0.0000	500	0.5160
0.8	0.0000	1000	0.6250
0.9	0.0000		
1.0	0.0000		

Feature: 10

Total shots returned:	141
Total shots with feature:	38
Total shots with feature returned:	16
Precision at shots with feature:	0.2105
Average precision:	0.0816



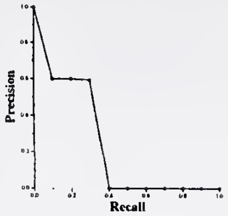
Interpolated recall	precision	Precision at n shots	
0.0	0.3333	5	0.2000
0.1	0.2632	10	0.2000
0.2	0.2162	15	0.1333
0.3	0.1368	20	0.2500
0.4	0.1143	30	0.2000
0.5	0.0000	100	0.1300
0.6	0.0000	200	0.0800
0.7	0.0000	500	0.0320
0.8	0.0000	1000	0.0160
0.9	0.0000		
1.0	0.0000		



Run score (dot) versus median (---) versus best (box) by feature

```
TREC 2002 Video Track
Feature Extraction Results

Run ID:
  CMU.r2

Processing type:
  automatic

System training type:
  A (with knowledge of test set)
```

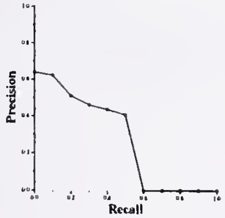
```
Feature:                               1

Total shots returned:                  539
Total shots with feature:              962
Total shots with feature returned:     318

Precision at shots with feature:       0.3306

Average precision:                     0.1766

          Interpolated           Precision at
          recall precision         n shots
           0.0  1.0000              5  0.2000
           0.1  0.5981             10  0.1000
           0.2  0.5981             15  0.2667
           0.3  0.5918             20  0.3000
           0.4  0.0000             30  0.3000
           0.5  0.0000            100  0.4100
           0.6  0.0000            200  0.5450
           0.7  0.0000            500  0.5880
           0.8  0.0000           1000  0.3180
           0.9  0.0000
           1.0  0.0000
```



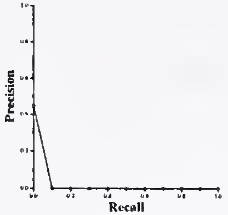
No submission for feature 2

No submission for feature 3

```
Feature:                               4

Total shots returned:                  690
Total shots with feature:              486
Total shots with feature returned:     277

Precision at shots with feature:       0.4239

Average precision:                     0.2738

          Interpolated           Precision at
          recall precision         n shots
           0.0  0.6400              5  0.2000
           0.1  0.6235             10  0.4000
           0.2  0.5116             15  0.4667
           0.3  0.4620             20  0.4500
           0.4  0.4365             30  0.5667
           0.5  0.4073            100  0.5800
           0.6  0.0000            200  0.5000
           0.7  0.0000            500  0.4200
           0.8  0.0000           1000  0.2770
           0.9  0.0000
           1.0  0.0000
```

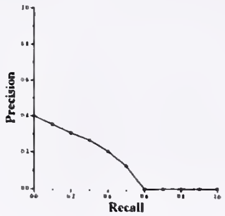


```
Feature:                               5

Total shots returned:                  117
Total shots with feature:              521
Total shots with feature returned:     49

Precision at shots with feature:       0.0940

Average precision:                     0.0357

          Interpolated           Precision at
          recall precision         n shots
           0.0  0.4483              5  0.4000
           0.1  0.0000             10  0.2000
           0.2  0.0000             15  0.3333
           0.3  0.0000             20  0.4000
           0.4  0.0000             30  0.4333
           0.5  0.0000            100  0.4000
           0.6  0.0000            200  0.2450
           0.7  0.0000            500  0.0980
           0.8  0.0000           1000  0.0490
           0.9  0.0000
           1.0  0.0000
```

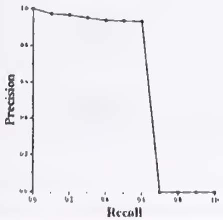


No submission for feature 6

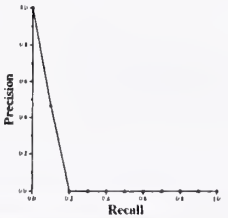
```
Feature:                               7

Total shots returned:                  731
Total shots with feature:              110
Total shots with feature returned:     59

Precision at shots with feature:       0.2545

Average precision:                     0.1301

          Interpolated           Precision at
          recall precision         n shots
           0.0  0.4000              5  0.4000
           0.1  0.3529             10  0.2000
           0.2  0.3056             15  0.2000
           0.3  0.2661             20  0.2500
           0.4  0.2047             30  0.3000
           0.5  0.1250            100  0.2700
           0.6  0.0000            200  0.2050
           0.7  0.0000            500  0.1100
           0.8  0.0000           1000  0.0590
           0.9  0.0000
           1.0  0.0000
```



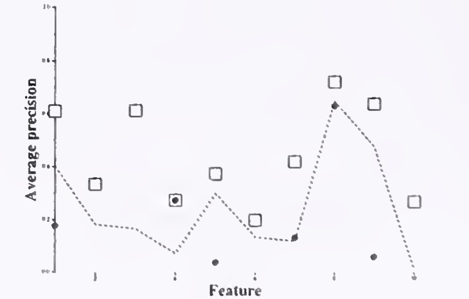
```
Feature:                               8

Total shots returned:                  1000
Total shots with feature:              1382
Total shots with feature returned:     924

Precision at shots with feature:       0.6686

Average precision:                     0.6304

          Interpolated           Precision at
          recall precision         n shots
           0.0  1.0000              5  1.0000
           0.1  0.9725             10  0.9000
           0.2  0.9673             15  0.8667
           0.3  0.9522             20  0.8500
           0.4  0.9391             30  0.8333
           0.5  0.9371            100  0.9500
           0.6  0.9340            200  0.9700
           0.7  0.0000            500  0.9500
           0.8  0.0000           1000  0.9240
           0.9  0.0000
           1.0  0.0000
```



```
Feature:                               9

Total shots returned:                  454
Total shots with feature:              1221
Total shots with feature returned:     210

Precision at shots with feature:       0.1720

Average precision:                     0.0568

          Interpolated           Precision at
          recall precision         n shots
           0.0  1.0000              5  0.8000
           0.1  0.4673             10  0.4000
           0.2  0.0000             15  0.2667
           0.3  0.0000             20  0.2500
           0.4  0.0000             30  0.2000
           0.5  0.0000            100  0.1300
           0.6  0.0000            200  0.1300
           0.7  0.0000            500  0.4200
           0.8  0.0000           1000  0.2100
           0.9  0.0000
           1.0  0.0000
```



No submission for feature 10



Run score (dot) versus median (---) versus best (box) by feature

TREC 2002 Video Track
Feature Extraction Results

Run ID:
  CMU.r3

Processing type:
  automatic

System training type:
  A (with knowledge of test set)

No submission for feature 1

No submission for feature 2

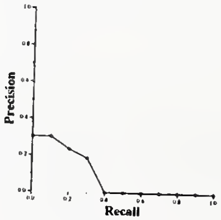
No submission for feature 3

No submission for feature 4

No submission for feature 5

No submission for feature 6

Feature:	7
Total shots returned:	317
Total shots with feature:	110
Total shots with feature returned:	40
Precision at shots with feature:	0.2273
Average precision:	0.0818



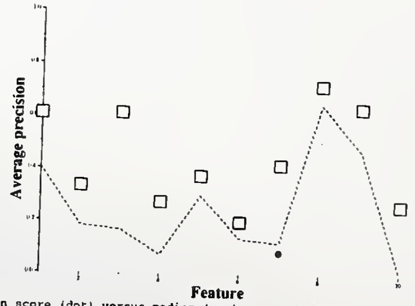
Interpolated recall	precision
0.0	0.3019
0.1	0.3019
0.2	0.2358
0.3	0.1862
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.0000
10	0.2000
15	0.2000
20	0.1500
30	0.2333
100	0.2200
200	0.1750
500	0.0800
1000	0.0400

No submission for feature 8

No submission for feature 9

No submission for feature 10



Run score (dot) versus median (---) versus best (box) by feature

TREC 2002 Video Track
Feature Extraction Results

Run ID:
CLIPS-LIT-GEOD

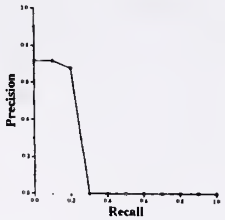
Processing type:
automatic

System training type:
B (no knowledge of test set)

No submission for feature 1

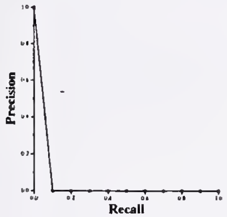
No submission for feature 2

Feature:	3
Total shots returned:	195
Total shots with feature:	415
Total shots with feature returned:	118
Precision at shots with feature:	0.2843
Average precision:	0.1782



Interpolated recall	precision	Precision at n shots	
0.0	0.7167	5	0.2000
0.1	0.7167	10	0.2000
0.2	0.6774	15	0.2667
0.3	0.0000	20	0.4500
0.4	0.0000	30	0.6000
0.5	0.0000	100	0.7000
0.6	0.0000	200	0.5900
0.7	0.0000	500	0.2360
0.8	0.0000	1000	0.1180
0.9	0.0000		
1.0	0.0000		

Feature:	4
Total shots returned:	25
Total shots with feature:	486
Total shots with feature returned:	18
Precision at shots with feature:	0.0370
Average precision:	0.0232



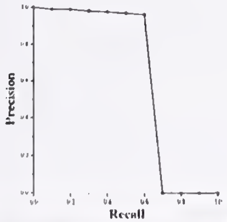
Interpolated recall	precision	Precision at n shots	
0.0	1.0000	5	0.2000
0.1	0.0000	10	0.5000
0.2	0.0000	15	0.6667
0.3	0.0000	20	0.7000
0.4	0.0000	30	0.6000
0.5	0.0000	100	0.1800
0.6	0.0000	200	0.0900
0.7	0.0000	500	0.0360
0.8	0.0000	1000	0.0180
9.9	0.0000		
1.0	0.0000		

No submission for feature 5

No submission for feature 6

No submission for feature 7

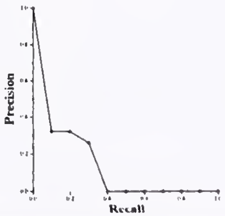
Feature:	8
Total shots returned:	1000
Total shots with feature:	1382
Total shots with feature returned:	924
Precision at shots with feature:	0.6686
Average precision:	0.6486



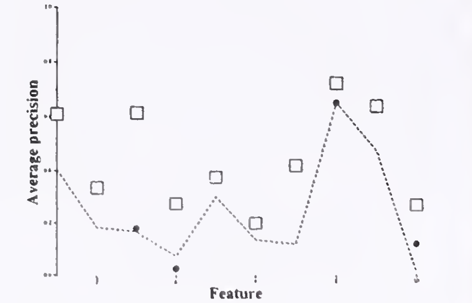
Interpolated recall	precision	Precision at n shots	
0.0	1.0000	5	0.6000
0.1	0.9904	10	0.8000
0.2	0.9904	15	0.8667
0.3	0.9814	20	0.9000
0.4	0.9770	30	0.9333
0.5	0.9762	100	0.9800
0.6	0.9625	200	0.9850
0.7	0.0000	500	0.9740
0.8	0.0000	1000	0.9240
0.9	0.0000		
1.0	0.0000		

No submission for feature 9

Feature:	10
Total shots returned:	70
Total shots with feature:	38
Total shots with feature returned:	14
Precision at shots with feature:	0.2895
Average precision:	0.1173



Interpolated recall	precision	Precision at n shots	
0.0	1.0000	5	0.2000
0.1	0.3214	10	0.2000
0.2	0.3214	15	0.2000
0.3	0.2600	20	0.2500
0.4	0.0000	30	0.3000
0.5	0.0000	100	0.1400
0.6	0.0000	200	0.0700
0.7	0.0000	500	0.0280
0.8	0.0000	1000	0.0140
0.9	0.0000		
1.0	0.0000		



Run score (dot) versus median (---) versus best (box) by feature

TREC 2002 Video Track
Feature Extraction Results

Run ID:
CLIPS-LIT-LIMSU

Processing type:
automatic

System training type:
B (no knowledge of test set)

No submission for feature 1

No submission for feature 2

No submission for feature 3

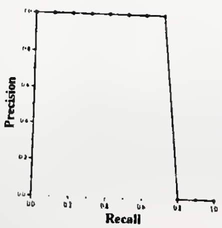
No submission for feature 4

No submission for feature 5

No submission for feature 6

No submission for feature 7

Feature:	8
Total shots returned:	1000
Total shots with feature:	1382
Total shots with feature returned:	997
Precision at shots with feature:	0.7214
Average precision:	0.7208

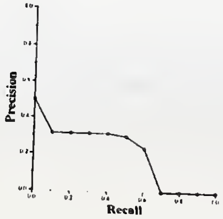


Interpolated recall	precision
0.0	1.0000
0.1	1.0000
0.2	1.0000
0.3	1.0000
0.4	1.0000
0.5	0.9987
0.6	0.9980
0.7	0.9583
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	1.0000
10	1.0000
15	1.0000
20	1.0000
30	1.0000
100	1.0000
200	1.0000
500	1.0000
1000	0.9970

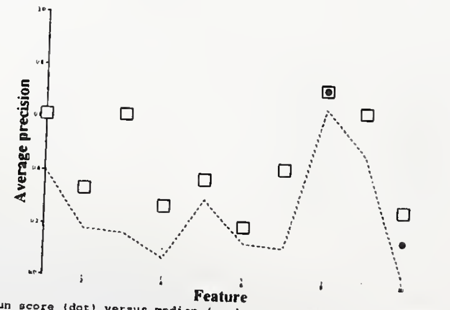
No submission for feature 9

Feature:	10
Total shots returned:	148
Total shots with feature:	38
Total shots with feature returned:	23
Precision at shots with feature:	0.2105
Average precision:	0.1490



Interpolated recall	precision
0.0	0.5000
0.1	0.3137
0.2	0.3137
0.3	0.3137
0.4	0.3137
0.5	0.2985
0.6	0.2247
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.2000
10	0.1000
15	0.1333
20	0.1000
30	0.1000
100	0.2300
200	0.1150
500	0.0460
1000	0.0230



Run score (dot) versus median (---) versus best (box) by feature

TREC 2002 Video Track
Feature Extraction Results

Run ID:
DCUFE2002

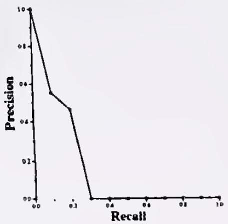
Processing type:
automatic

System training type:
B (no knowledge of test set)

No submission for feature 1

No submission for feature 2

Feature:	3
Total shots returned:	300
Total shots with feature:	415
Total shots with feature returned:	114
Precision at shots with feature:	0.2747
Average precision:	0.1543



Interpolated recall	precision
0.0	1.0000
0.1	0.5506
0.2	0.4620
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	1.0000
10	0.8000
15	0.8667
20	0.8500
30	0.6333
100	0.5300
200	0.4400
500	0.2280
1000	0.1140

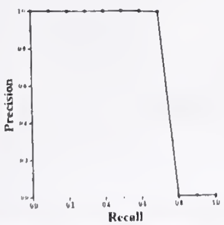
No submission for feature 4

No submission for feature 5

No submission for feature 6

No submission for feature 7

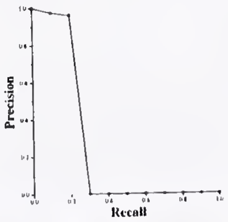
Feature:	8
Total shots returned:	1000
Total shots with feature:	1382
Total shots with feature returned:	987
Precision at shots with feature:	0.7142
Average precision:	0.7103



Interpolated recall	precision
0.0	1.0000
0.1	1.0000
0.2	0.9968
0.3	0.9958
0.4	0.9958
0.5	0.9958
0.6	0.9928
0.7	0.9870
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	1.0000
10	1.0000
15	1.0000
20	1.0000
30	1.0000
100	1.0000
200	0.9950
500	0.9940
1000	0.9870

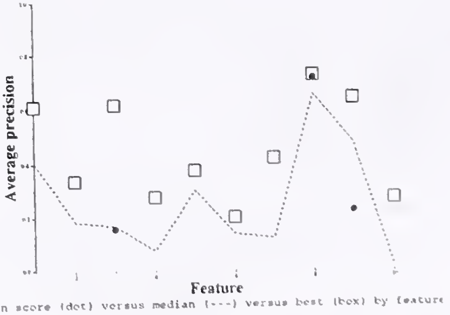
Feature:	9
Total shots returned:	300
Total shots with feature:	1221
Total shots with feature returned:	281
Precision at shots with feature:	0.2301
Average precision:	0.2221



Interpolated recall	precision
0.0	1.0000
0.1	0.9756
0.2	0.9612
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	1.0000
10	1.0000
15	1.0000
20	0.9500
30	0.9667
100	0.9700
200	0.8650
500	0.5620
1000	0.2810

No submission for feature 10



Run score (dot) versus median (---) versus best (box) by feature

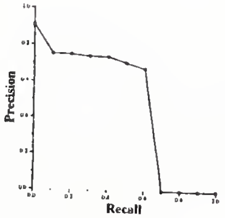
TREC 2002 Video Track
Feature Extraction Results

Run ID:
Eurecom1

Processing type:
automatic

System training type:
B (no knowledge of test set)

Feature:	1
Total shots returned:	942
Total shots with feature:	962
Total shots with feature returned:	624
Precision at shots with feature:	0.6486
Average precision:	0.4710

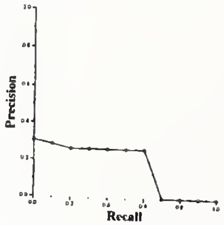


Interpolated recall	precision
0.0	0.9091
0.1	0.7524
0.2	0.7500
0 3	0.7397
0.4	0.7377
0.5	0.7053
0.6	0.6744
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	0.8000
10	0.9000
15	0.8667
20	0.8000
30	0.7667
100	0.7700
200	0.7450
500	0.7360
1000	0.6240

No submission for feature 2

Feature:	3
Total shots returned:	979
Total shots with feature:	415
Total shots with feature returned:	255
Precision at shots with feature:	0.2313
Average precision:	0.1499

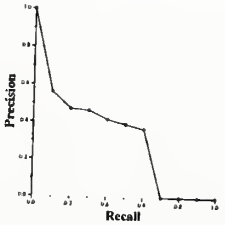


Interpolated recall	precision
0.0	0.3056
0.1	0.2857
0.2	0.2615
0.3	0.2615
0.4	0.2615
0.5	0.2615
0.6	0.2615
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	0.0000
10	0.0000
15	0.0000
20	0.0500
30	0.1000
100	0.2800
200	0.2500
500	0.2300
1000	0.2550

No submission for feature 4

Feature:	5
Total shots returned:	945
Total shots with feature:	521
Total shots with feature returned:	339
Precision at shots with feature:	0.4146
Average precision:	0.3031

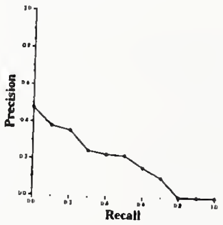


Interpolated recall	precision
0.0	1.0000
0.1	0.5625
0.2	0.4726
0.3	0.4645
0 4	0.4170
0.5	0.3913
0.6	0.3667
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	1.0000
10	0.8000
15	0,7333
20	0.7000
30	0.6000
100	0.5400
200	0.4700
500	0.4140
1000	0.3390

No submission for feature 6

Feature:	7
Total shots returned:	967
Total shots with feature:	110
Total shots with feature returned:	83
Precision at shots with feature:	0.2545
Average precision:	0.1837



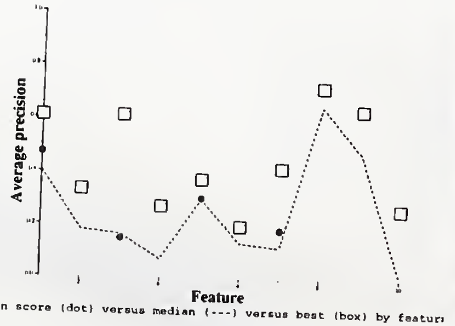
Interpolated recall	precision
0.0	0.4737
0.1	0.3793
0.2	0.3548
0.3	0.2464
0.4	0.2278
0,5	0.2218
0.6	0.1568
0.7	0.1034
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	0.4000
10	0.3000
15	0.4000
20	0.4500
30	0.3667
100	0 2700
200	0.2100
500	0.1420
1000	0.0830

No submission for feature 8

No submission for feature 9

No submission for feature 10



Run score (dot) versus median (---) versus best (box) by featur

TREC 2002 Video Track
Feature Extraction Results

Run ID:
Fudan_FE_Sys1

Processing type:
automatic

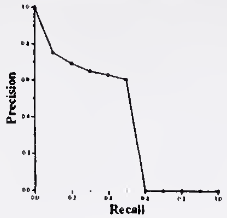
System training type:
B (no knowledge of test set)

Feature: 1

Total shots returned: 1000
Total shots with feature: 962
Total shots with feature returned: 574

Precision at shots with feature: 0.5780

Average precision: 0.3947



Interpolated recall	precision
0.0	1.0000
0.1	0.7519
0.2	0.6918
0.3	0.6494
0.4	0.6294
0.5	0.6034
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

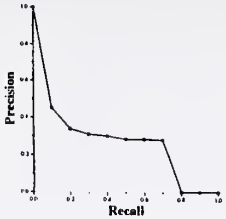
Precision at n shots	
5	0.6000
10	0.7000
15	0.8000
20	0.7500
30	0.7333
100	0.7800
200	0.7000
500	0.6400
1000	0.5740

Feature: 2

Total shots returned: 1000
Total shots with feature: 351
Total shots with feature returned: 271

Precision at shots with feature: 0.3020

Average precision: 0.2432



Interpolated recall	precision
0.0	1.0000
0.1	0.4516
0.2	0.3377
0.3	0.3101
0.4	0.2996
0.5	0.2824
0.6	0.2824
0.7	0.2777
0.8	0.0000
0.9	0.0000
1.0	0.0000

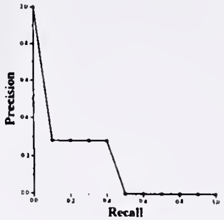
Precision at n shots	
5	0.4000
10	0.4000
15	0.2667
20	0.3000
30	0.3333
100	0.4200
200	0.3300
500	0.2940
1000	0.2710

Feature: 3

Total shots returned: 680
Total shots with feature: 415
Total shots with feature returned: 190

Precision at shots with feature: 0.2386

Average precision: 0.1112



Interpolated recall	precision
0.0	1.0000
0.1	0.2794
0.2	0.2794
0.3	0.2794
0.4	0.2794
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

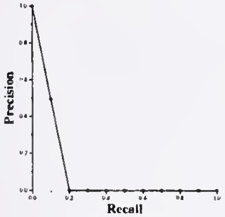
Precision at n shots	
5	0.6000
10	0.3000
15	0.2000
20	0.2500
30	0.2333
100	0.1700
200	0.1750
500	0.2560
1000	0.1900

Feature: 4

Total shots returned: 178
Total shots with feature: 486
Total shots with feature returned: 83

Precision at shots with feature: 0.1708

Average precision: 0.0711



Interpolated recall	precision
0.0	1.0000
0.1	0.4965
0.2	0.0000
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

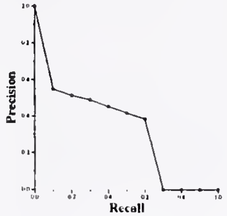
Precision at n shots	
5	0.2000
10	0.4000
15	0.3333
20	0.2500
30	0.2333
100	0.4400
200	0.4150
500	0.1660
1000	0.0830

Feature: 5

Total shots returned: 1000
Total shots with feature: 521
Total shots with feature returned: 354

Precision at shots with feature: 0.4376

Average precision: 0.3247



Interpolated recall	precision
0.0	1.0000
0.1	0.5484
0.2	0.5146
0.3	0.4909
0.4	0.4534
0.5	0.4188
0.6	0.3848
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

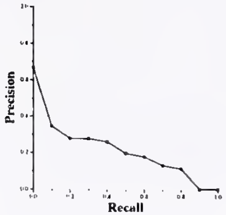
Precision at n shots	
5	0.6000
10	0.7000
15	0.6000
20	0.7000
30	0.6333
100	0.5300
200	0.5100
500	0.4480
1000	0.3540

Feature: 6

Total shots returned: 1000
Total shots with feature: 127
Total shots with feature returned: 107

Precision at shots with feature: 0.2756

Average precision: 0.1978



Interpolated recall	precision
0.0	0.6667
0.1	0.3469
0.2	0.2784
0.3	0.2771
0.4	0.2600
0.5	0.1963
0.6	0.1777
0.7	0.1303
0.8	0.1121
0.9	0.0000
1.0	0.0000

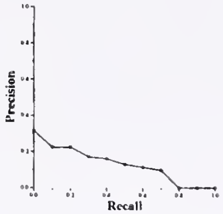
Precision at n shots	
5	0.4000
10	0.3000
15	0.2667
20	0.4000
30	0.3333
100	0.2700
200	0.2600
500	0.1580
1000	0.1070

Feature: 7

Total shots returned: 1000
Total shots with feature: 110
Total shots with feature returned: 84

Precision at shots with feature: 0.2182

Average precision: 0.1177



Interpolated recall	precision
0.0	0.3158
0.1	0.2233
0.2	0.2233
0.3	0.1706
0.4	0.1606
0.5	0.1294
0.6	0.1134
0.7	0.0981
0.8	0.0000
0.9	0.0000
1.0	0.0000

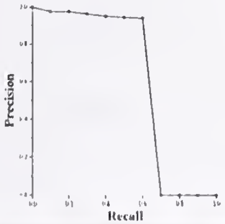
Precision at n shots	
5	0.0000
10	0.0000
15	0.2000
20	0.3000
30	0.2333
100	0.2200
200	0.1600
500	0.1260
1000	0.0840

Feature: 8

Total shots returned: 1000
Total shots with feature: 1382
Total shots with feature returned: 932

Precision at shots with feature: 0.6744

Average precision: 0.6448



Interpolated recall	precision
0.0	1.0000
0.1	0.9768
0.2	0.9768
0.3	0.9640
0.4	0.9516
0.5	0.9457
0.6	0.9410
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

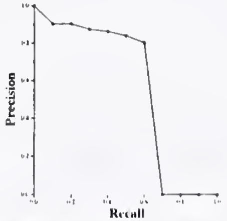
Precision at n shots	
5	1.0000
10	0.9000
15	0.9333
20	0.9000
30	0.9333
100	0.9700
200	0.9700
500	0.9540
1000	0.9320

Feature: 9

Total shots returned: 917
Total shots with feature: 1221
Total shots with feature returned: 735

Precision at shots with feature: 0.6020

Average precision: 0.5211



Interpolated recall	precision
0.0	1.0000
0.1	0.9034
0.2	0.9034
0.3	0.8744
0.4	0.8629
0.5	0.8401
0.6	0.8024
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

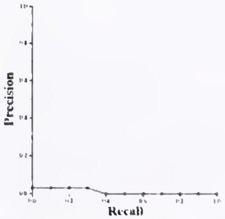
Precision at n shots	
5	1.0000
10	0.9000
15	0.9333
20	0.9500
30	0.9333
100	0.8600
200	0.8800
500	0.8620
1000	0.7350

Feature: 10

Total shots returned: 533
Total shots with feature: 38
Total shots with feature returned: 14

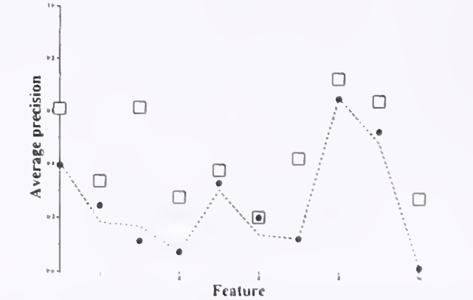
Precision at shots with feature: 0.0000

Average precision: 0.0081



Interpolated recall	precision
0.0	0.0296
0.1	0.0296
0.2	0.0296
0.3	0.0296
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.0000
10	0.0000
15	0.0000
20	0.0000
30	0.0000
100	0.0100
200	0.0050
500	0.0280
1000	0.0140



Run score (dot) versus median (---) versus best (box) by feature

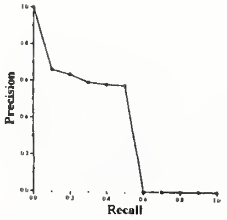
TREC 2002 Video Track
Feature Extraction Results

Run ID:
Fudan_FE_Sys2

Processing type:
automatic

System training type:
B (no knowledge of test set)

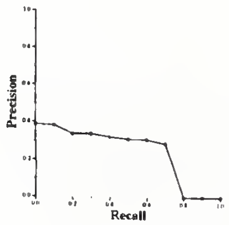
Feature:	1
Total shots returned:	1000
Total shots with feature:	962
Total shots with feature returned:	556
Precision at shots with feature:	0.5572
Average precision:	0.3530



Interpolated recall	precision
0.0	1.0000
0.1	0.6599
0.2	0.6331
0.3	0.5907
0.4	0.5793
0.5	0 5728
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.6000
10	0.6000
15	0.7333
20	0.7500
30	0.8000
100	0.6600
200	0.6400
500	0.5860
1000	0.5560

Feature:	2
Total shots returned:	1000
Total shots with feature:	351
Total shots with feature returned:	274
Precision at shots with feature:	0.3276
Average precision:	0.2456



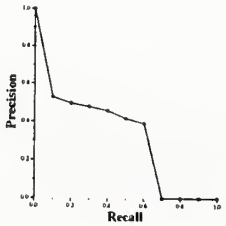
Interpolated recall	precision
0.0	0.3864
0.1	0.3814
0.2	0.3369
0.3	0.3369
0.4	0.3224
0.5	0.3105
0.6	0.3089
0.7	0.2874
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.0000
10	0.2000
15	0.2000
20	0.3000
30	0.3000
100	0.3700
200	0.3250
500	0.3060
1000	0.2740

No submission for feature 3

No submission for feature 4

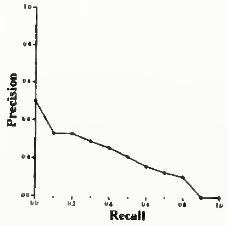
Feature:	5
Total shots returned:	1000
Total shots with feature:	521
Total shots with feature returned:	355
Precision at shots with feature:	0.4357
Average precision:	0.3162



Interpolated recall	precision
0.0	1.0000
0.1	0.5289
0.2	0.4964
0.3	0.4787
0.4	0.4577
0.5	0.4167
0.6	0.3917
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.6000
10	0.8000
15	0.6667
20	0.6000
30	0.5000
100	0.4800
200	0.4300
500	0.4440
1000	0.3550

Feature:	6
Total shots returned:	1000
Total shots with feature:	127
Total shots with feature returned:	103
Precision at shots with feature:	0.2913
Average precision:	0.1896

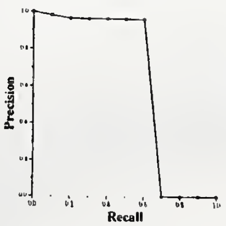


Interpolated recall	precision
0.9	0.5000
0.1	0.3291
0.2	0.3291
0.3	0.2887
0.4	0.2550
0.5	0.2098
0.6	0.1601
0.7	0.1292
0.8	0.1052
0.9	0.0000
1.C	0.0000

Precision at n shots	
5	0 4000
10	0.3000
15	0.3333
20	0.2500
30	0.3000
100	0.3100
200	0.2550
500	0.1560
1000	0.1030

No submission for feature 7

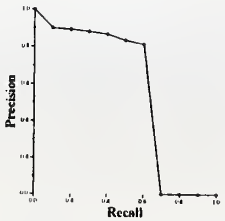
Feature:	8
Total shots returned:	1000
Total shots with feature:	1382
Total shots with feature returned:	951
Precision at shots with feature:	0.6881
Average precision:	0.6632



Interpolated recall	precision
0.0	1.0000
0 1	0.9817
0.2	0.9644
0.3	0.9624
0.4	0.9621
0 5	0.9621
0.6	0.9603
0.7	0.0000
0 8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	1.0000
10	1.0000
15	0.9333
20	0.9500
30	0.9667
100	0.9800
200	0.9800
500	0.9600
1000	0.9510

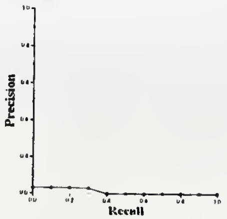
Feature:	9
Total shots returned:	1000
Total shots with feature:	1221
Total shots with feature returned:	799
Precision at shots with feature:	0.6544
Average precision:	0.5638



Interpolated recall	precision
0.0	1.0000
0.1	0.9004
0.2	0.8931
0.3	0.8826
0.4	0.8687
0.5	0.8360
0.6	0.8147
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

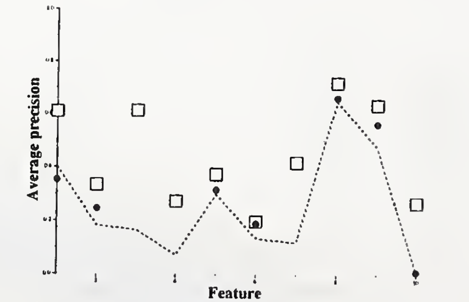
Precision at n shots	
5	1.0000
10	0.9000
15	0.9333
20	0.9500
30	0.9333
100	0.8500
200	0.8900
500	0.8780
1000	0.7990

Feature:	10
Total shots returned:	528
Total shots with feature:	38
Total shots with feature returned:	14
Precision at shots with feature:	0.0000
Average precision:	0.0087



Interpolated recall	precision
0.0	0.0310
0.1	0.0310
0.2	0.0310
0.3	0.0282
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.0000
10	0.0000
15	0.0000
20	0.0000
30	0.0000
100	0.0100
200	0.0100
500	0.0280
1000	0.0140



Run score (dot) versus median (---) versus best (box) by feature

TREC 2002 Video Track
Feature Extraction Results

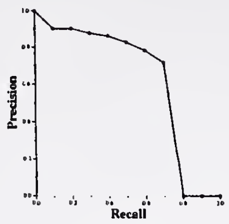
Run ID:
IBM-1

Processing type:
automatic

System training type:
B (no knowledge of test set)

Feature: 1

Total shots returned:	1000
Total shots with feature:	962
Total shots with feature returned:	702
Precision at shots with feature:	0.7131
Average precision:	0.6091

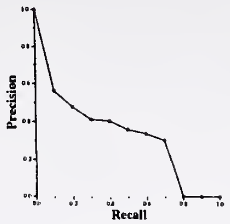


Interpolated recall	precision
0.0	1.0000
0.1	0.9020
0.2	0.9020
0.3	0.8771
0.4	0.8616
0.5	0.8265
0.6	0.7821
0.7	0.7162
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.4000
10	0.7000
15	0.7333
20	0.8000
30	0.8333
100	0.8800
200	0.8950
500	0.8520
1000	0.7020

Feature: 2

Total shots returned:	1000
Total shots with feature:	351
Total shots with feature returned:	278
Precision at shots with feature:	0.3989
Average precision:	0.3348

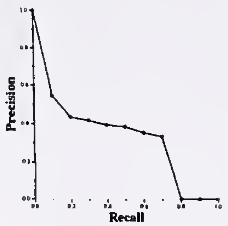


Interpolated recall	precision
0.0	1.0000
0.1	0.5625
0.2	0.4778
0.3	0.4113
0.4	0.4038
0.5	0.3580
0.6	0.3369
0.7	0.3022
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.8000
10	0.7000
15	0.6667
20	0.6500
30	0.6667
100	0.5100
200	0.4550
500	0.3580
1000	0.2780

Feature: 3

Total shots returned:	1000
Total shots with feature:	415
Total shots with feature returned:	312
Precision at shots with feature:	0.3880
Average precision:	0.3271

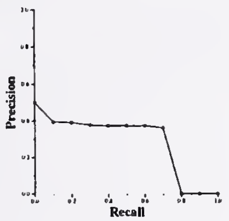


Interpolated recall	precision
0.0	1.0000
0.1	0.5455
0.2	0.4343
0.3	0.4167
0.4	0.3932
0.5	0.3838
0.6	0.3527
0.7	0.3326
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	1.0000
10	0.9000
15	0.8000
20	0.7000
30	0.7000
100	0.5100
200	0.4300
500	0.3800
1000	0.3120

Feature: 4

Total shots returned:	1000
Total shots with feature:	486
Total shots with feature returned:	361
Precision at shots with feature:	0.3642
Average precision:	0.2707

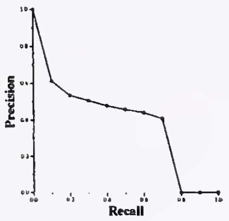


Interpolated recall	precision
0.0	0.5000
0.1	0.3963
0.2	0.3931
0.3	0.3798
0.4	0.3752
0.5	0.3752
0.6	0.3748
0.7	0.3632
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.2000
10	0.3000
15	0.3333
20	0.3000
30	0.3000
100	0.3100
200	0.3850
500	0.3640
1000	0.3610

Feature: 5

Total shots returned:	1000
Total shots with feature:	521
Total shots with feature returned:	391
Precision at shots with feature:	0.4702
Average precision:	0.3744

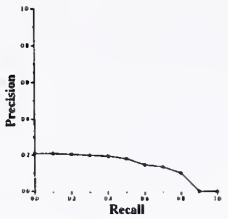


Interpolated recall	precision
0.0	1.0000
0.1	0.6136
0.2	0.5347
0.3	0.5076
0.4	0.4787
0.5	0.4592
0.6	0.4418
0.7	0.4101
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.4000
10	0.7000
15	0.6667
20	0.6000
30	0.5667
100	0.5900
200	0.5300
500	0.4680
1000	0.3910

Feature: 6

Total shots returned:	1000
Total shots with feature:	127
Total shots with feature returned:	102
Precision at shots with feature:	0.1969
Average precision:	0.1337

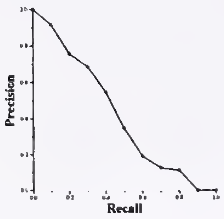


Interpolated recall	precision
0.0	0.2101
0.1	0.2101
0.2	0.2055
0.3	0.2000
0.4	0.1955
0.5	0.1818
0.6	0.1482
0.7	0.1366
0.8	0.1025
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.0000
10	0.0000
15	0.0000
20	0.0500
30	0.1333
100	0.2000
200	0.1950
500	0.1520
1000	0.1020

Feature: 7

Total shots returned:	1000
Total shots with feature:	110
Total shots with feature returned:	96
Precision at shots with feature:	0.4636
Average precision:	0.4181

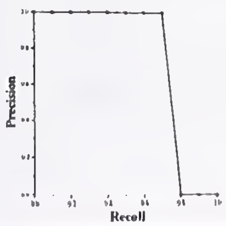


Interpolated recall	precision
0.0	1.0000
0.1	0.9167
0.2	0.7586
0.3	0.6875
0.4	0.5476
0.5	0.3481
0.6	0.1919
0.7	0.1259
0.8	0.1125
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	1.0000
10	0.9000
15	0.8667
20	0.8500
30	0.7333
100	0.5000
200	0.2850
500	0.1440
1000	0.0960

Feature: 8

Total shots returned:	1000
Total shots with feature:	1382
Total shots with feature returned:	990
Precision at shots with feature:	0.7164
Average precision:	0.7127



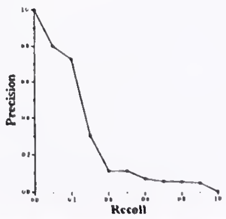
Interpolated recall	precision
0.0	1.0000
0.1	0.9981
0.2	0.9981
0.3	0.9981
0.4	0.9969
0.5	0.9935
0.6	0.9918
0.7	0.9900
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	1.0000
10	1.0000
15	1.0000
20	1.0000
30	1.0000
100	0.9900
200	0.9950
500	0.9980
1000	0.9900

Because of a clerical error in the submission, the intended system output was available too late for an official evalution.
See the site report for results.

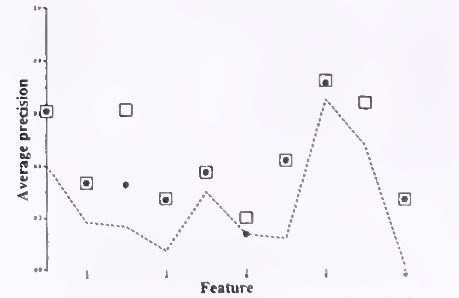
Feature: 10

Total shots returned:	1000
Total shots with feature:	38
Total shots with feature returned:	37
Precision at shots with feature:	0.2895
Average precision:	0.2675



Interpolated recall	precision
0.0	1.0000
0.1	0.8000
0.2	0.7273
0.3	0.3023
0.4	0.1099
0.5	0.1099
0.6	0.0695
0.7	0.0550
0.8	0.0518
0.9	0.0457
1.0	0.0000

Precision at n shots	
5	0.8000
10	0.7000
15	0.6000
20	0.5000
30	0.3333
100	0.1400
200	0.1050
500	0.0540
1000	0.0370



Run score (dot) versus median (---) versus best (box) by feature

TREC 2002 Video Track
Feature Extraction Results

Run ID:
IBM-2

Processing type:
automatic

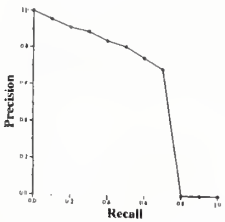
System training type:
B (no knowledge of test set)

Feature: 1

Total shots returned: 1000
Total shots with feature: 962
Total shots with feature returned: 678

Precision at shots with feature: 0.6850

Average precision: 0.6002



Interpolated recall	precision
0.0	1.0000
0.1	0.9558
0.2	0 9112
0.3	0 8873
0.4	0.8370
0.5	0.8057
0.6	0.7439
0 7	0.6839
0 8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	1.0000
10	1.0000
15	1.0000
20	1.0000
30	1.0000
100	0.9500
200	0.9100
500	0.8220
1000	0.6780

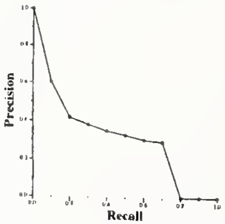
No submission for feature 2

Feature: 3

Total shots returned: 1000
Total shots with feature: 415
Total shots with feature returned: 293

Precision at shots with feature: 0.3663

Average precision: 0.2883



Interpolated recall	precision
0.0	1.0000
0.1	0.6027
0.2	0.4189
0.3	0.3832
0.4	0.3502
0.5	0.3281
0.6	0.3041
0.7	0.2933
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.6000
10	0.6000
15	0.6667
20	0.6000
30	0.6000
100	0.5300
200	0.4150
500	0.3460
1000	0.2930

No submission for feature 4

No submission for feature 5

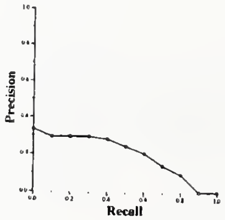
No submission for feature 6

Feature: 7

Total shots returned: 1000
Total shots with feature: 110
Total shots with feature returned: 88

Precision at shots with feature: 0.2818

Average precision: 0.1814



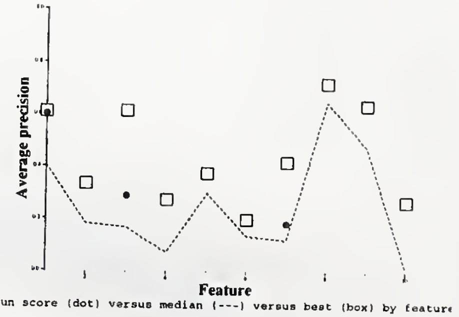
Interpolated recall	precision
0.0	0.3333
0.1	0.2931
0.2	0.2931
0.3	0.2931
0.4	0.2812
0.5	0.2423
0.6	0.2050
0.7	0.1385
0.8	0.0925
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.2000
10	0.1000
15	0.1333
20	0.2000
30	0.2333
100	0.2700
200	0.2550
500	0.1520
1000	0.0880

No submission for feature 8

No submission for feature 9

No submission for feature 10



Run score (dot) versus median (---) versus best (box) by feature

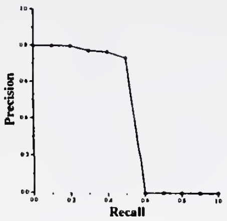
TREC 2002 Video Track
Feature Extraction Results

Run ID:
MediaMill1

Processing type:
automatic

System training type:
B (no knowledge of test set)

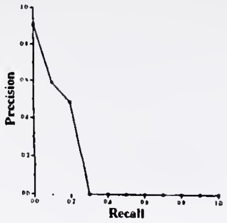
Feature:	1
Total shots returned:	744
Total shots with feature:	962
Total shots with feature returned:	525
Precision at shots with feature:	0.5457
Average precision:	0.4030



Interpolated recall	precision
0.0	0.7973
0.1	0.7973
0.2	0.7948
0.3	0.7692
0.4	0.7613
0.5	0.7277
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.0000
10	0.3000
15	0.5333
20	0.6500
30	0.6233
100	0.6700
200	0.7900
500	0.7620
1000	0.5250

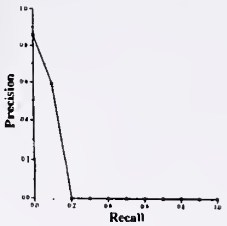
Feature:	2
Total shots returned:	213
Total shots with feature:	351
Total shots with feature returned:	92
Precision at shots with feature:	0.2621
Average precision:	0.1595



Interpolated recall	precision
0.0	0.9091
0.1	0.5902
0.2	0.4815
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.8000
10	0.9000
15	0.8667
20	0.8500
30	0.7667
100	0.4900
200	0.4550
500	0.1840
1000	0.0920

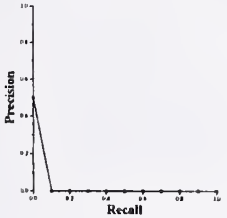
Feature:	3
Total shots returned:	101
Total shots with feature:	415
Total shots with feature returned:	55
Precision at shots with feature:	0.1325
Average precision:	0.0886



Interpolated recall	precision
0.0	0.8571
0.1	0.5914
0.2	0.0000
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.8000
10	0.8000
15	0.8000
20	0.6500
30	0.7333
100	0.5500
200	0.2750
500	0.1100
1000	0.0550

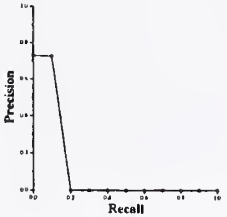
Feature:	4
Total shots returned:	35
Total shots with feature:	486
Total shots with feature returned:	12
Precision at shots with feature:	0.0247
Average precision:	0.0079



Interpolated recall	precision
0.0	0.5000
0.1	0.0000
0.2	0.0000
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.2000
10	0.2000
15	0.2000
20	0.3000
30	0.3333
100	0.1200
200	0.0600
500	0.0240
1000	0.0120

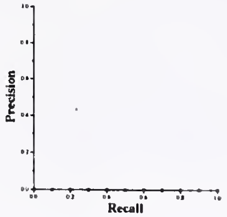
Feature:	5
Total shots returned:	133
Total shots with feature:	521
Total shots with feature returned:	90
Precision at shots with feature:	0.1727
Average precision:	0.1141



Interpolated recall	precision
0.0	0.7288
0.1	0.7253
0.2	0.0000
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.4000
10	0.3000
15	0.5333
20	0.6500
30	0.6333
100	0.7100
200	0.4500
500	0.1800
1000	0.0900

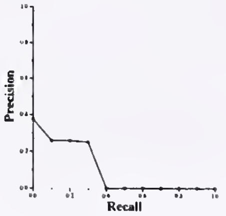
Feature:	6
Total shots returned:	4
Total shots with feature:	127
Total shots with feature returned:	0
Precision at shots with feature:	0.0000
Average precision:	0.0000



Interpolated recall	precision
0.0	0.0000
0.1	0.0000
0.2	0.0000
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.0000
10	0.0000
15	0.0000
20	0.0000
30	0.0000
100	0.0000
200	0.0000
500	0.0000
1000	0.0000

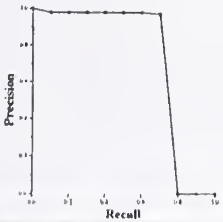
Feature:	7
Total shots returned:	166
Total shots with feature:	110
Total shots with feature returned:	36
Precision at shots with feature:	0.2364
Average precision:	0.0785



Interpolated recall	precision
0.0	0.3750
0.1	0.2588
0.2	0.2586
0.3	0.2500
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.0000
10	0.3000
15	0.2667
20	0.2000
30	0.2333
100	0.2300
200	0.1800
500	0.0720
1000	0.0360

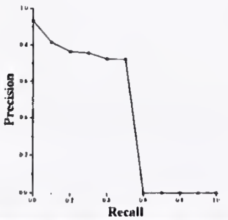
Feature:	8
Total shots returned:	1000
Total shots with feature:	1382
Total shots with feature returned:	970
Precision at shots with feature:	0.7019
Average precision:	0.6809



Interpolated recall	precision
0.0	1.0000
0.1	0.9794
0.2	0.9794
0.3	0.9794
0.4	0.9794
0.5	0.9794
0.6	0.9786
0.7	0.9709
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.8000
10	0.9000
15	0.9333
20	0.9000
30	0.9333
100	0.9600
200	0.9700
500	0.9760
1000	0.9700

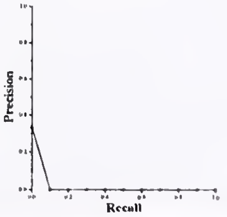
Feature:	9
Total shots returned:	1000
Total shots with feature:	1221
Total shots with feature returned:	716
Precision at shots with feature:	0.5864
Average precision:	0.4378



Interpolated recall	precision
0.0	0.9328
0.1	0.8150
0.2	0.7639
0.3	0.7565
0.4	0.7235
0.5	0.7214
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

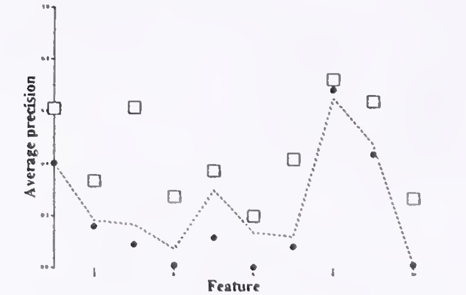
Precision at n shots	
5	0.4000
10	0.4000
15	0.5333
20	0.6500
30	0.7667
100	0.9200
200	0.7500
500	0.7540
1000	0.7160

Feature:	10
Total shots returned:	6
Total shots with feature:	38
Total shots with feature returned:	1
Precision at shots with feature:	0.0263
Average precision:	0.0088



Interpolated recall	precision
0.0	0.3333
0.1	0.0000
0.2	0.0000
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.2000
10	0.1000
15	0.0667
20	0.0500
30	0.0333
100	0.0100
200	0.0050
500	0.0020
1000	0.0010



Run score (dot) versus median (---) versus best (box) by feature

TREC 2002 Video Track
Feature Extraction Results

Run ID:
MediaMill2

Processing type:
automatic

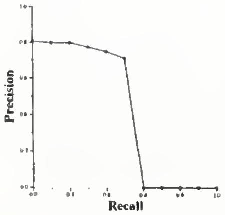
System training type:
B (no knowledge of test set)

Feature: 1

Total shots returned: 775
Total shots with feature: 962
Total shots with feature returned: 525

Precision at shots with feature: 0.5457

Average precision: 0.4016



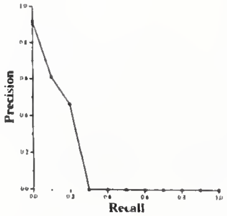
Interpolated recall	precision	Precision at n shots	
0.0	0.8077	5	0.0000
0.1	0.7984	10	0.2000
0.2	0.7984	15	0.4667
0.3	0.7746	20	0.6000
0.4	0.7481	30	0.6667
0.5	0.7107	100	0.7500
0.6	0.0000	200	0.7650
0.7	0.0000	500	0.7520
0.8	0.0000	1000	0.5250
0.9	0.0000		
1.0	0.0000		

Feature: 2

Total shots returned: 211
Total shots with feature: 351
Total shots with feature returned: 87

Precision at shots with feature: 0.2479

Average precision: 0.1538



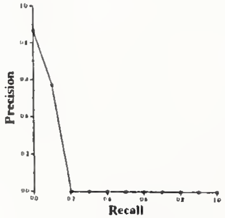
Interpolated recall	precision	Precision at n shots	
0.0	0.9167	5	0.8000
0.1	0.6102	10	0.9000
0.2	0.4630	15	0.8667
0.3	0.0000	20	0.8500
0.4	0.0000	30	0.8000
0.5	0.0000	100	0.5300
0.6	0.0000	200	0.4350
0.7	0.0000	500	0.1740
0.8	0.0000	1000	0.0870
0.9	0.0000		
1.0	0.0000		

Feature: 3

Total shots returned: 107
Total shots with feature: 415
Total shots with feature returned: 57

Precision at shots with feature: 0.1373

Average precision: 0.0907



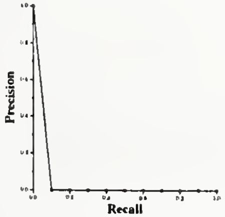
Interpolated recall	precision	Precision at n shots	
0.0	0.8667	5	0.8000
0.1	0.5714	10	0.8000
0.2	0.0000	15	0.8667
0.3	0.0000	20	0.7000
0.4	0.0000	30	0.7667
0.5	0.0000	100	0.5600
0.6	0.0000	200	0.2850
0.7	0.0000	500	0.1140
0.8	0.0000	1000	0.0570
0.9	0.0000		
1.0	0.0000		

Feature: 4

Total shots returned: 29
Total shots with feature: 486
Total shots with feature returned: 10

Precision at shots with feature: 0.0206

Average precision: 0.0084



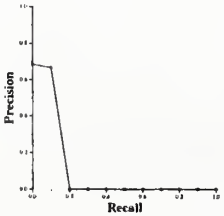
Interpolated recall	precision	Precision at n shots	
0.0	1.0000	5	0.2000
0.1	0.0000	10	0.3000
0.2	0.0000	15	0.3333
0.3	0.0000	20	0.3500
0.4	0.0000	30	0.3333
0.5	0.0000	100	0.1000
0.6	0.0000	200	0.0500
0.7	0.0000	500	0.0200
0.8	0.0000	1000	0.0100
0.9	0.0000		
1.0	0.0000		

Feature: 5

Total shots returned: 170
Total shots with feature: 521
Total shots with feature returned: 86

Precision at shots with feature: 0.1651

Average precision: 0.0978



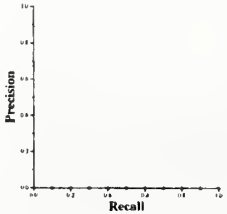
Interpolated recall	precision	Precision at n shots	
0.0	0.6857	5	0.4000
0.1	0.6667	10	0.5000
0.2	0.0000	15	0.4000
0.3	0.0000	20	0.5000
0.4	0.0000	30	0.6333
0.5	0.0000	100	0.6400
0.6	0.0000	200	0.4300
0.7	0.0000	500	0.1720
0.8	0.0000	1000	0.0860
0.9	0.0000		
1.0	0.0000		

Feature: 6

Total shots returned: 4
Total shots with feature: 127
Total shots with feature returned: 0

Precision at shots with feature: 0.0000

Average precision: 0.0000



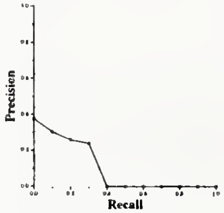
Interpolated recall	precision	Precision at n shots	
0.0	0.0000	5	0.0000
0.1	0.0000	10	0.0000
0.2	0.0000	15	0.0000
0.3	0.0000	20	0.0000
0.4	0.0000	30	0.0000
0.5	0.0000	100	0.0000
0.6	0.0000	200	0.0000
0.7	0.0000	500	0.0000
0.8	0.0000	1000	0.0000
0.9	0.0000		
1.0	0.0000		

Feature: 7

Total shots returned: 173
Total shots with feature: 110
Total shots with feature returned: 35

Precision at shots with feature: 0.2455

Average precision: 0.0819



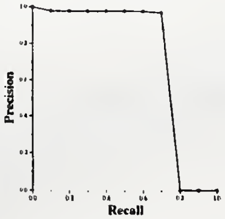
Interpolated recall	precision	Precision at n shots	
0.0	0.3750	5	0.0000
0.1	0.3023	10	0.3000
0.2	0.2584	15	0.2667
0.3	0.2391	20	0.2000
0.4	0.0000	30	0.2667
0.5	0.0000	100	0.2500
0.6	0.0000	200	0.1750
0.7	0.0000	500	0.0700
0.8	0.0000	1000	0.0350
0.9	0.0000		
1.0	0.0000		

Feature: 8

Total shots returned: 1000
Total shots with feature: 1382
Total shots with feature returned: 970

Precision at shots with feature: 0.7019

Average precision: 0.6809



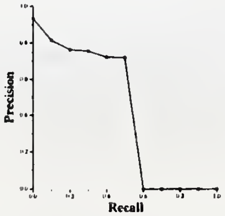
Interpolated recall	precision	Precision at n shots	
0.0	1.0000	5	0.8000
0.1	0.9794	10	0.9000
0.2	0.9794	15	0.9333
0.3	0.9794	20	0.9000
0.4	0.9794	30	0.9333
0.5	0.9794	100	0.9600
0.6	0.9786	200	0.9700
0.7	0.9709	500	0.9760
0.8	0.0000	1000	0.9700
0.9	0.0000		
1.0	0.0000		

Feature: 9

Total shots returned: 1000
Total sho:s with feature: 1221
Total shots with feature returned: 716

Precision at shots with feature: 0.5864

Average precision: 0.4378



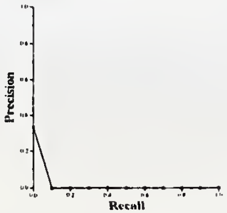
Interpolated recall	precision	Precision at n shots	
0.0	0.9328	5	0.4000
0.1	0.8150	10	0.4000
0.2	0.7639	15	0.5333
0.3	0.7565	20	0.6500
0.4	0.7235	30	0.7667
0.5	0.7214	100	0.9200
0.6	0.0000	200	0.7900
0.7	0.0000	500	0.7540
0.8	0.0000	1000	0.7160
0.9	0.0000		
1.0	0.0000		

Feature: 10

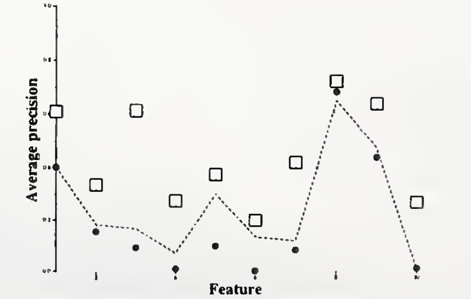
Total shots returned: 5
Total shots with feature: 38
Total shots with feature returned: 1

Precision at shots with feature: 0.0263

Average precision: 0.0088



Interpolated recall	precision	Precision at n shots	
0.0	0.3333	5	0.2000
0.1	0.0000	10	0.1000
0.2	0.0000	15	0.0667
0.3	0.0000	20	0.0500
0.4	0.0000	30	0.0333
0.5	0.0000	100	0.0100
0.6	0.0000	200	0.0050
0.7	0.0000	500	0.0020
0.8	0.0000	1000	0.0010
0.9	0.0000		
1.0	0.0000		



Run score (dot) versus median (---) versus best (box) by feature

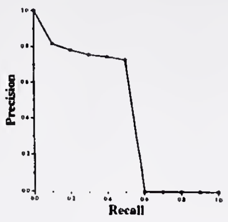
TREC 2002 Video Track
Feature Extraction Results

Run ID:
MSRA

Processing type:
automatic

System training type:
B (no knowledge of test set)

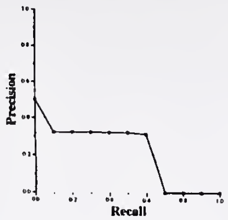
Feature:	1
Total shots returned:	800
Total shots with feature:	962
Total shots with feature returned:	570
Precision at shots with feature:	0.5925
Average precision:	0.4555



Interpolated recall	precision
0.0	1.0000
0.1	0.8169
0.2	0.7811
0.3	0.7558
0.4	0.7448
0.5	0.7277
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	1.0000
10	1.0000
15	0.9333
20	0.9500
30	0.8667
100	0.7900
200	0.8100
500	0.7420
1000	0.5700

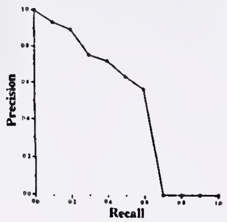
Feature:	2
Total shots returned:	800
Total shots with feature:	351
Total shots with feature returned:	241
Precision at shots with feature:	0.3191
Average precision:	0.2055



Interpolated recall	precision
0.0	0.5000
0.1	0.3220
0.2	0.3220
0.3	0.3220
0.4	0.3200
0.5	0.3200
0.6	0.3114
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	0.4000
10	0.3000
15	0.2667
20	0.2500
30	0.2000
100	0.2500
200	0.2750
500	0.3160
1000	0.2410

Feature:	3
Total shots returned:	450
Total shots with feature:	415
Total shots with feature returned:	253
Precision at shots with feature:	0.5735
Average precision:	0.4729

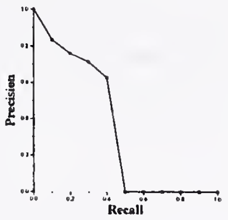


Interpolated recall	precision
0.0	1.0000
0.1	0.9333
0.2	0.8925
0.3	0.7529
0.4	0.7198
0.5	0.6341
0.6	0.5625
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	1.0000
10	0.9000
15	0.9333
20	0.9500
30	0.9667
100	0.8600
200	0.7300
500	0.5060
1000	0.2530

No submission for feature 4

Feature:	5
Total shots returned:	400
Total shots with feature:	521
Total shots with feature returned:	245
Precision at shots with feature:	0.4702
Average precision:	0.3479

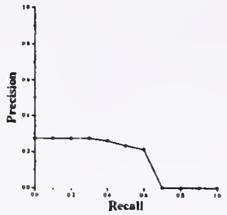


Interpolated recall	precision
0.0	1.0000
0.1	0.8358
0.2	0.7609
0.3	0.7156
0.4	0.6287
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	1.0000
10	1.0000
15	0.7333
20	0.8000
30	0.7667
100	0.7900
200	0.7200
500	0.4900
1000	0.2450

No submission for feature 6

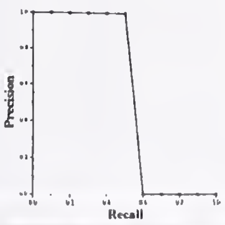
Feature:	7
Total shots returned:	500
Total shots with feature:	110
Total shots with feature returned:	75
Precision at shots with feature:	0.2364
Average precision:	0.1512



Interpolated recall	precision
0.0	0.2748
0.1	0.2748
0.2	0.2748
0.3	0.2748
0.4	0.2619
0.5	0.2350
0.6	0.2141
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	0.2000
10	0.1000
15	0.0667
20	0.0500
30	0.1000
100	0.2100
200	0.2500
500	0.1500
1000	0.0750

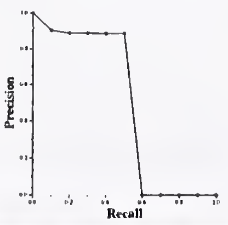
Feature:	8
Total shots returned:	800
Total shots with feature:	1382
Total shots with feature returned:	792
Precision at shots with feature:	0.5731
Average precision:	0.5702



Interpolated recall	precision
0.0	1.0000
0.1	1.0000
0.2	0.9968
0.3	0.9953
0.4	0.9933
0.5	0.9905
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	1.0000
10	1.0000
15	1.0000
20	1.0000
30	1.0000
100	1.0000
200	1.0000
500	0.9920
1000	0.7920

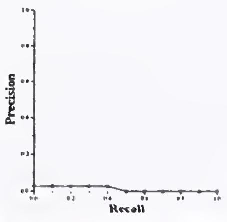
Feature:	9
Total shots returned:	800
Total shots with feature:	1221
Total shots with feature returned:	709
Precision at shots with feature:	0.5807
Average precision:	0.5115



Interpolated recall	precision
0.0	1.0000
0.1	0.9050
0.2	0.8807
0.3	0.8807
0.4	0.8866
0.5	0.8866
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

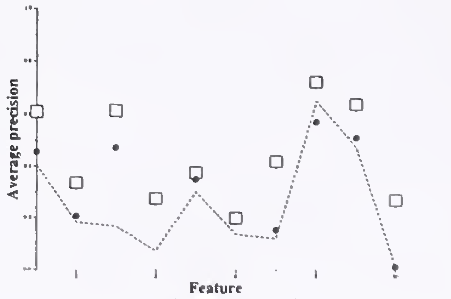
Precision at n	shots
5	1.0000
10	0.7000
15	0.8000
20	0.8000
30	0.8000
100	0.9000
200	0.8950
500	0.8860
1000	0.7090

Feature:	10
Total shots returned:	600
Total shots with feature:	38
Total shots with feature returned:	16
Precision at shots with feature:	0.0000
Average precision:	0.0086



Interpolated recall	precision
0.0	0.0274
0.1	0.0274
0.2	0.0274
0.3	0.0274
0.4	0.0274
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	0.0000
10	0.0000
15	0.0000
20	0.0000
30	0.0000
100	0.0000
200	0.0100
500	0.0240
1000	0.0160



Run score (dot) versus median (---) versus best (box) by feature

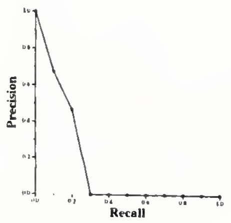
TREC 2002 Video Track
Feature Extraction Results

Run ID:
  TZI_univ_bremen

Processing type:
  automatic

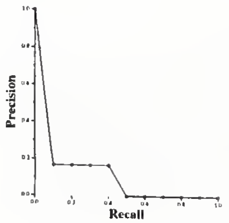
System training type:
  B (no knowledge of test set)

```
Feature:                                  1

Total shots returned:                     553
Total shots with feature:                 962
Total shots with feature returned:        255

Precision at shots with feature:          0,2651

Average precision:                        0.1497
```



```
Interpolated            Precision at
recall) precision         n shcts
 0.0  1.0000               5  1.0000
 0.1  0.6736              10  0.8000
 0.2  0.4653              15  0.6667
 0.3  0.0000              20  0.6000
 0.4  0.0000              30  0.6667
 0.5  0.0000             100  0.6300
 0.6  0.0000             200  0.6000
 0.7  0.0000             500  0.4620
 0 8  0.0000            1000  0.2550
 0.9  0.0000
 1.0  0.0000
```

```
Feature:                                  2

Total shots returned:                     936
Total shots with feature:                 351
Total shots with feature returned:        156

Precision at shots with feature:          0.1026

Average precision:                        0.0660
```



```
Interpolated            Precision at
recall precisicn          n shota
 0.0  1.0000               5  0.6000
 0.1  0.1674              10  0.3000
 0.2  0.1674              15  0.2667
 0.3  0 1674              20  0.2000
 0.4  0.1674              30  0.1667
 0.5  0.0000             100  0.1200
 0.6  0.0000             200  0.0900
 0.7  0.0000             500  0.1120
 0.8  0 0000            1000  0.1560
 0.9  0.0000
 1.0  0.0000
```

No submission for feature 3

No submission for feature 4

No submission for feature 5

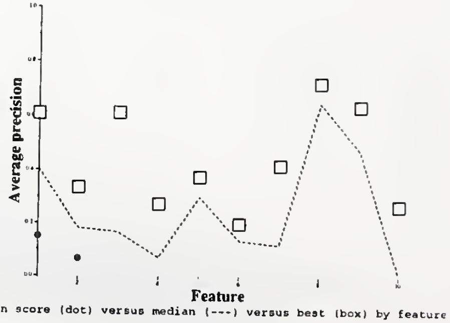
No submission for feature 6

No submission for feature 7

No submission for feature 8

No submission for feature 9

No submission for feature 10



Run score (dot) versus median (---) versus best (box) by feature

TREC 2002 Video Track
Feature Extraction Results

Run ID:
  UMD1

Processing type:
  automatic

System training type:
  B (no knowledge of test set)

No submission for feature 1

No submission for feature 2

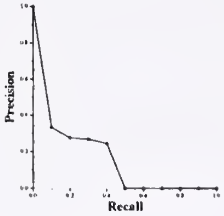
No submission for feature 3

No submission for feature 4

No submission for feature 5

No submission for feature 6

Feature:	7
Total shots returned:	200
Total shots with feature:	110
Total shots with feature returned:	47
Precision at shots with feature:	0.2091
Average precision:	0.1167



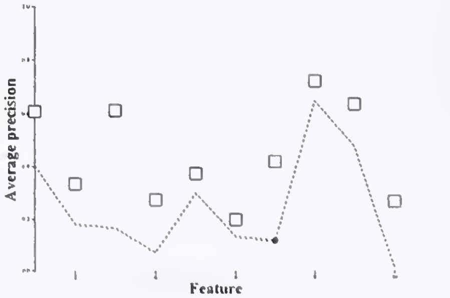
Interpolated recall	precision
0.0	1.0000
0.1	0.3333
0.2	0.2750
0.3	0.2676
0.4	0.2444
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.2000
10	0.3000
15	0.2000
20	0.2000
30	0.1667
100	0.2200
200	0.2350
500	0.0940
1000	0.0470

No submission for feature 8

No submission for feature 9

No submission for feature 10



Run score (dot) versus median (---) versus best (box) by feature

TREC 2002 Video Track
Feature Extraction Results

Run ID:
UnivO_MT1

Processing type:
automatic

System training type:
B (no knowledge of test set)

No submission for feature 1

No submission for feature 2

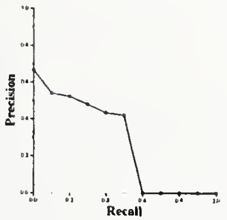
No submission for feature 3

Feature: 4

Total shots returned: 600
Total shots with feature: 486
Total shots with feature returned: 251

Precision at shots with feature: 0.4259

Average precision: 0.2480



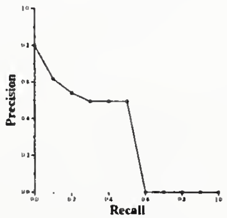
Interpolated		Precision at	
recall	precision	n	shots
0.0	0 6667	5	0.6000
0.1	0.5400	10	0.6000
0.2	0.5234	15	0.4667
0.3	0.4806	20	0.4500
0.4	0.4352	30	0.5000
0.5	0.4209	100	0.5400
0.6	0.0000	200	0.5200
0.7	0.0000	500	0.4220
0.8	0.0000	1000	0.2510
0.9	0.0000		
1.0	0.0000		

Feature: 5

Total shots returned: 600
Total shots with feature: 521
Total shots with feature returned: 294

Precision at shots with feature: 0.4875

Average precision: 0.2988



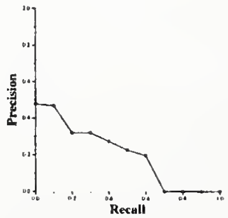
Interpolated		Precision at	
recall	precision	n	shots
0.0	0.8000	5	0.8000
0.1	0.6170	10	0.6000
0.2	0.5404	15	0.7333
0.3	0.4944	20	0.7000
0.4	0.4944	30	0.6667
0 5	0.4944	100	0.6000
0.6	0.0000	200	0.5350
0.7	0.0000	500	0.4880
0.8	0.0000	1000	0.2940
0.9	0.0000		
1.0	0.0000		

Feature: 6

Total shots returned: 600
Total shots with feature: 127
Total shots with feature returned: 86

Precision at shots with feature: 0.3071

Average precision: 0.1929



Interpolated		Precision at	
recall	precision	n	shots
0.0	0 4783	5	0 4000
0.1	0.4688	10	0.4000
0.2	0.3206	15	0.4667
0.3	0.3206	20	0.4000
0.4	0.2742	30	0.4333
0.5	0.2273	100	0.2900
0.6	0.1959	200	0.2600
0.7	0.0000	500	0.1640
0.8	0.0000	1000	0.0860
0 9	0.0000		
1.0	0.0000		

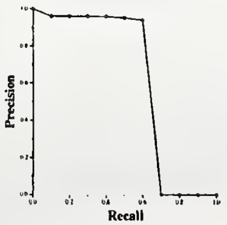
No submission for feature 7

Feature: 8

Total shots returned: 1000
Total shots with feature: 1382
Total shots with feature returned: 934

Precision at shots with feature: 0.6758

Average precision: 0.6448



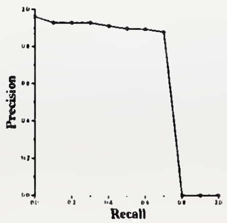
Interpolated		Precision at	
recall	precision	n	shots
0.0	1.0000	5	1.0000
0.1	0.9615	10	1.0000
0.2	0.9615	15	1.0000
0.3	0.9615	20	1.0000
0.4	0.9608	30	1.0000
0.5	0.9527	100	0.9500
0.6	0.9414	200	0.9550
0.7	0.0000	500	0.9600
0.8	0.0000	1000	0.9340
0.9	0.0000		
1.0	0.0000		

Feature: 9

Total shots returned: 1000
Total shots with feature: 1221
Total shots with feature returned: 877

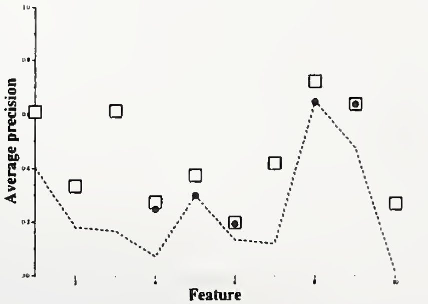
Precision at shots with feature: 0.7183

Average precision: 0.6370



Interpolated		Precision at	
recall	precision	n	shots
0.0	0.9615	5	0.8000
0.1	0.9286	10	0.9000
0.2	0.9286	15	0.9333
0.3	0.9279	20	0.9500
0.4	0.9106	30	0.8333
0.5	0.8964	100	0.8400
0.6	0.8929	200	0.8700
0.7	0.8779	500	0.9120
0.8	0.0000	1000	0.8770
0.9	0.0000		
1.0	0.0000		

No submission for feature 10



Run score (dot) versus median (---) versus best (box) by feature

TREC 2002 Video Track
Feature Extraction Results

Run ID:
UnivO_MT2

Processing type:
automatic

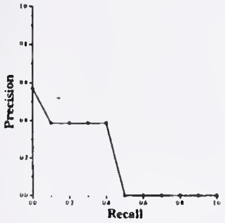
System training type:
B (no knowledge of test set)

No submission for feature 1

No submission for feature 2

No submission for feature 3

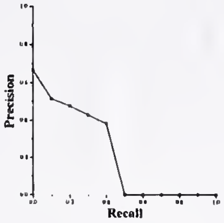
Feature:	4
Total shots returned:	600
Total shots with feature:	486
Total shots with feature returned:	223
Precision at shots with feature:	0.3827
Average precision:	0.1682



Interpolated recall	precision
0.0	0.5714
0.1	0.3874
0.2	0.3874
0.3	0.3874
0.4	0.3868
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.4000
10	0.4000
15	0.4000
20	0.4000
30	0.4000
100	0.2700
200	0.3500
500	0.3840
1000	0.2230

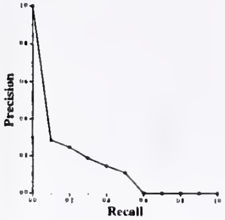
Feature:	5
Total shots returned:	600
Total shots with feature:	521
Total shots with feature returned:	228
Precision at shots with feature:	0.3839
Average precision:	0.1974



Interpolated recall	precision
0.0	0.6667
0.1	0.5161
0.2	0.4773
0.3	0.4286
0.4	0.3830
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.6000
10	0.4000
15	0.5333
20	0.5000
30	0.5000
100	0.4800
200	0.4900
500	0.3880
1000	0.2280

Feature:	6
Total shots returned:	600
Total shots with feature:	127
Total shots with feature returned:	65
Precision at shots with feature:	0.2441
Average precision:	0.1276

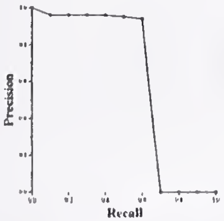


Interpolated recall	precision
0.0	1.0000
0.1	0.2857
0.2	0.2479
0.3	0.1893
0.4	0.1474
0.5	0.1109
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.6000
10	0.3000
15	0.4000
20	0.4000
30	0.3333
100	0.2400
200	0.1850
500	0.1160
1000	0.0650

No submission for feature 7

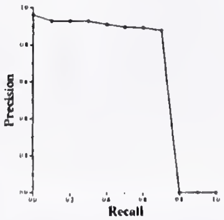
Feature:	8
Total shots returned:	1000
Total shots with feature:	1382
Total shots with feature returned:	934
Precision at shots with feature:	0.6758
Average precision:	0.6448



Interpolated recall	precision
0.0	1.0000
0.1	0.9615
0.2	0.9615
0.3	0.9615
0.4	0.9608
0.5	0.9527
0.6	0.9414
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	1.0000
10	1.0000
15	1.0000
20	1.0000
30	1.0000
100	0.9500
200	0.9550
500	0.9600
1000	0.9340

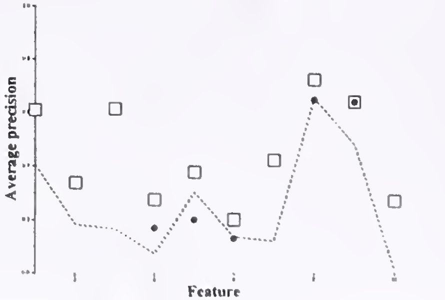
Feature:	9
Total shots returned:	1000
Total shots with feature:	1221
Total shots with feature returned:	877
Precision at shots with feature:	0.7183
Average precision:	0.6370



Interpolated recall	precision
0.0	0.9615
0.1	0.9286
0.2	0.9286
0.3	0.9279
0.4	0.9106
0.5	0.8964
0.6	0.8929
0.7	0.8779
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.8000
10	0.9000
15	0.9333
20	0.9500
30	0.8333
100	0.8400
200	0.8700
500	0.9120
1000	0.8770

No submission for feature 10



Run score (dot) versus median (---) versus best (box) by feature

TREC 2002 Video Track Runs (cont.)

Search

Interactive

*** A = system development with knowledge of search test set**

B = system development without

ID	**Training***	**Priority**	**Affiliation**
IuVf1	A	1	Indiana University
CMUInfInt1	B	1	Carnegie Mellon Univ.
CMU_INTERACTIVE_2	B	1	
DCUTrec11B.1	B	1	Dublin City University
DCUTrec11B.3	B	3	
DCUTrec11C.2	B	2	
DCUTrec11C.4	B	4	
IBM-4	B	4	IBM Research
ICMKM-1	B	1	Imperial College
MSRA.Q-Video.1	B	1	Microsoft Research Asia
MSRA.Q-Video.2.A	B	1	
UMDIqtrec	B	1	University of Maryland
UnivO_MT5	B	1	University of Oulu

Manual

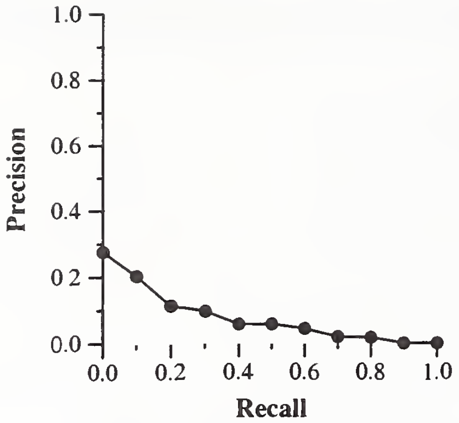
CLIPS+ASR	A	2	CLIPS-IMAG Lab
CLIPS+ASR+X	A	1	
CLIPS+X	A	3	
CMU_MANUAL1	A	1	Carnegie Mellon Univ.
CMU_MANUAL2	A	4	
CMU_MANUAL3	A	5	
Fudan_Search_Sys1	B	1	Fudan University
Fudan_Search_Sys2	B	2	
Fudan_Search_Sys3	B	3	
Fudan_Search_Sys4	B	4	
IBM-1	B	1	IBM Research
IBM-2	B	2	
IBM-3	B	3	
ICMKM-2	B	2	Imperial College
ICMKM-3	B	3	
ICMKM-4	B	4	
LL10_T	B	2	Lowlands team
LL10_TIac	B	1	
LL10_TIsc	B	3	
LL10_TIscG	B	4	
MSRA.Q-Video.3	B	1	Microsoft Research Asia
ProusSci	B	1	Prous Science
UMDMNAqt	B	3	University of Maryland
UMDMqtrec	B	2	
UnivO_MT1	B	2	University of Oulu
UnivO_MT2	B	3	
UnivO_MT3	B	4	

Topic descriptions

- 75 - Find shots with Eddie Rickenbacker in them. (2 image examples, 2 video examples)
- 76 - Find additional shots with James H. Chandler. (3 video examples)
- 77 - Find pictures of George Washington. (1 image example, 1 video example)
- 78 - Find shots with a depiction of Abraham Lincoln. (1 image example, 1 video example)
- 79 - Find shots of people spending leisure time at the beach, for example: walking, swimming, sunning, playing in the sand. Some part of the beach or buildings on it should be visible. (4 video examples)
- 80 - Find shots of one or more musicians: a man or woman playing a music instrument with instrumental music audible. Musician(s) and instrument(s) must be at least partly visible sometime during the shot. (2 video examples
- 81 - Find shots of football players (4 video examples)
- 82 - Find shots of one or more women standing in long dresses. Dress should be one piece and extend below knees. The entire dress from top to end of dress below knees should be visible at some point. (3 video examples)
- 83 - Find shots of the Golden Gate Bridge. (5 image examples)
- 84 - Find shots of Price Tower, designed by Frank Lloyd Wright and built in Bartlesville, Oklahoma. (1 image example)
- 85 - Find shots containing Washington Square Park's arch in New York City. The entire arch should be visible at some point. (1 video example)
- 86 - Find overhead views of cities - downtown and suburbs. The viewpoint should be higher than the highest building visible. (4 video examples)
- 87 - Find shots of oil fields, rigs, derricks, oil drilling/pumping equipment. Shots just of refineries are not desired. (1 video example)
- 88 - Find shots with a map (sketch or graphic) of the continental US. (4 video examples)
- 89 - Find shots of a living butterfly. (2 image examples)
- 90 - Find more shots with one or more snow-covered moutain peaks or ridges. Some sky must be visible them behind. (3 video examples)
- 91 - Find shots with one or more parrots (1 image example, 1 video example)
- 92 - Find shots with one or more sailboats, sailing ships, clipper ships, or tall ships - with some sail(s) unfurled. (4 image examples, 2 video examples)
- 93 - Find shots about live beef or dairy cattle, individual cows or bulls, herds of cattle. (5 video examples)
- 94 - Find more shots of one or more groups of people, a crowd, walking in an urban environment (for example with streets, traffic, and/or buildings) (3 video examples)
- 95 - Find shots of a nuclear explosion with a mushroom cloud. (3 video examples)
- 96 - Find additional shots with one or more US flags flapping. (2 video examples)
- 97 - Find more shots with microscopic views of living cells. (2 video examples)
- 98 - Find shots with a locomotive (and attached railroad cars if any) approaching the viewer. (5 video examples)
- 99 - Find shots of a rocket or missile taking off. Simulations are acceptible. (2 video examples)

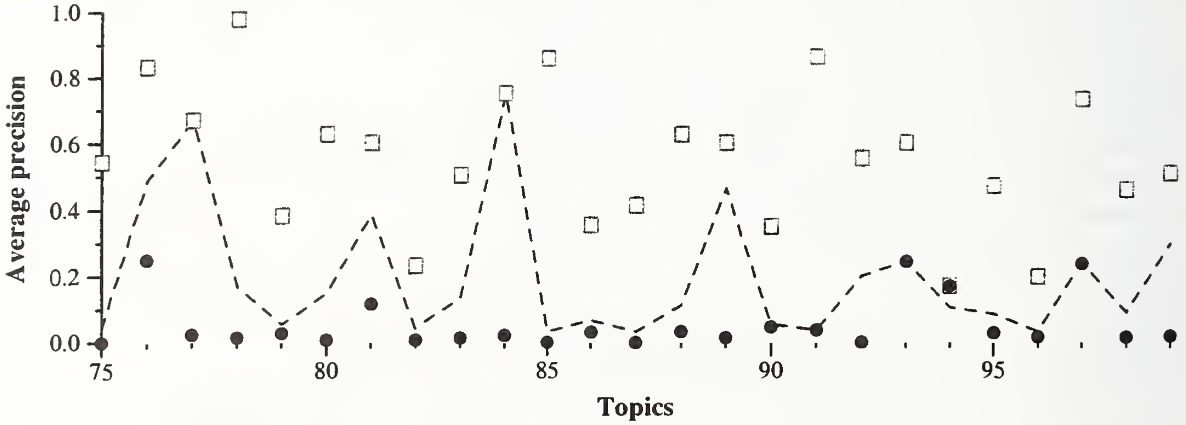
TREC 2002 Video Track: search results

Run ID:	IuVf1
Processing type:	interactive
System training type:	A (with knowledge of test set)
Topics summarized:	25
Total shots returned:	2065
Total relevant shots:	1445
Total relevant shots returned:	307
Precision at total relevant shots:	0.1030
Mean average precision:	0.0548

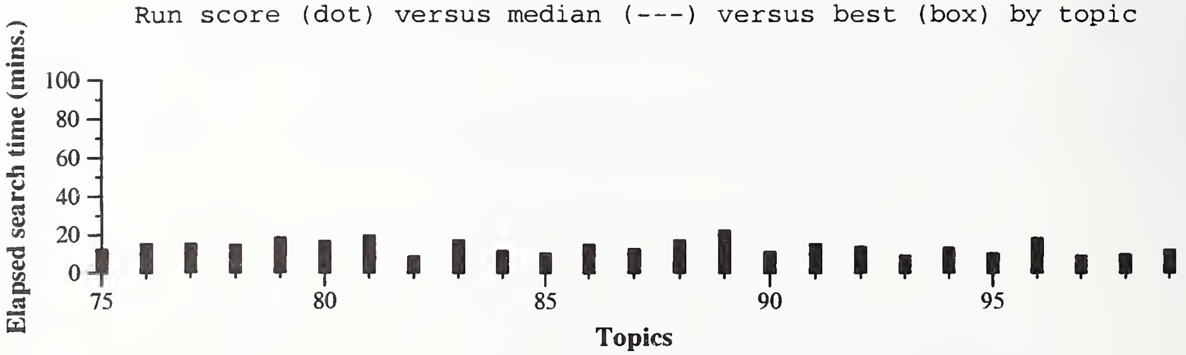


Interpolated recall	precision
0.0	0.2762
0.1	0.2028
0.2	0.1129
0.3	0.0955
0.4	0.0574
0.5	0.0574
0.6	0.0435
0.7	0.0189
0.8	0.0171
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.0640
10	0.1000
15	0.1040
20	0.1220
30	0.1427
100	0.1228



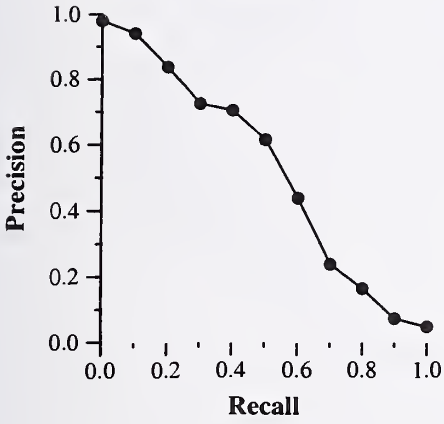
Run score (dot) versus median (---) versus best (box) by topic



Elapsed search time by topic

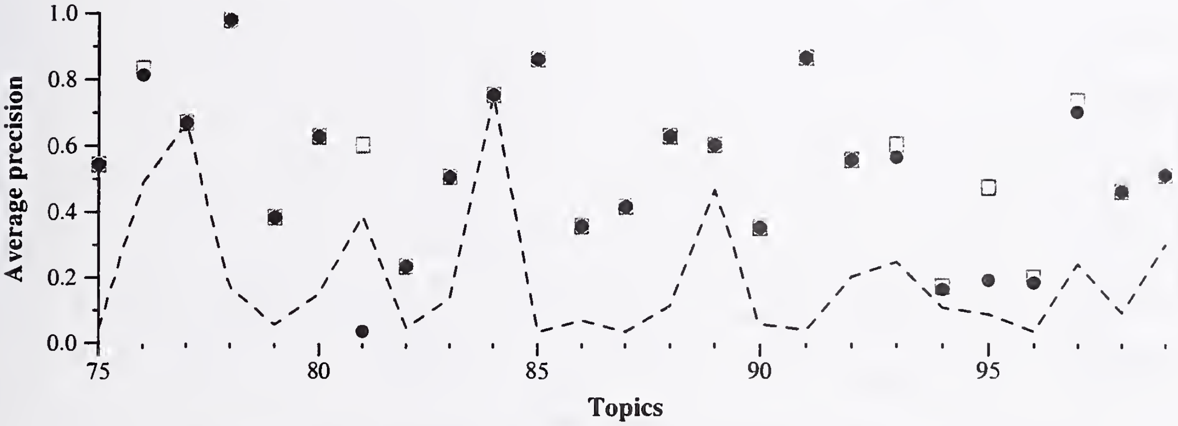
TREC 2002 Video Track: search results

Run ID:	CMUInfInt1
Processing type:	interactive
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	2255
Total relevant shots:	1445
Total relevant shots returned:	662
Precision at total relevant shots:	0.5358
Mean average precision:	0.5161

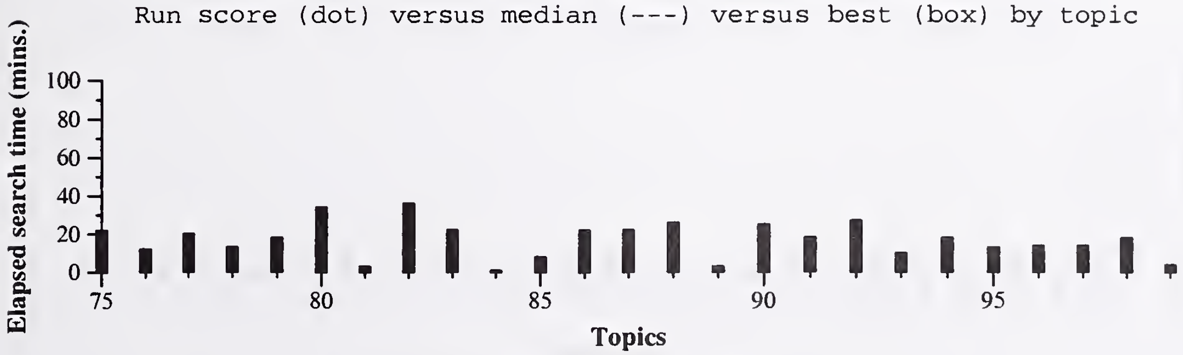


Interpolated recall	precision
0.0	0.9800
0.1	0.9402
0.2	0.8354
0.3	0.7217
0.4	0.7016
0.5	0.6119
0.6	0.4350
0.7	0.2358
0.8	0.1619
0.9	0.0705
1.0	0.0453

Precision at n shots	
5	0.8560
10	0.7600
15	0.6747
20	0.6200
30	0.5427
100	0.2648



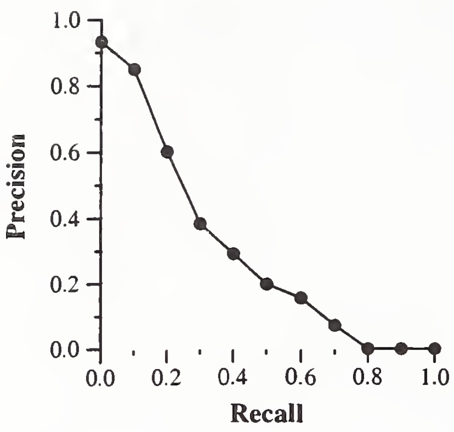
Run score (dot) versus median (---) versus best (box) by topic



Elapsed search time by topic

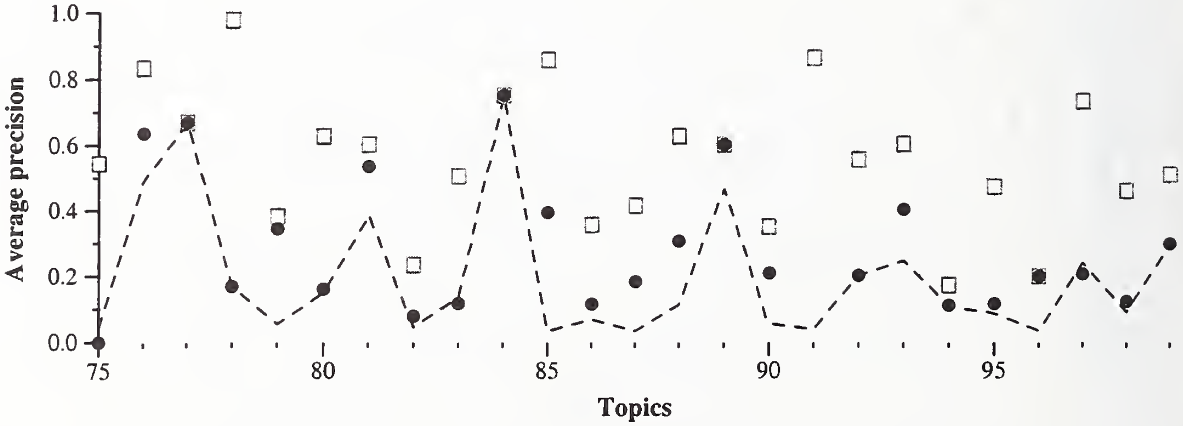
TREC 2002 Video Track: search results

Run ID:	CMU_INTERACTIVE_2
Processing type:	interactive
System training type:	B (no knowledge of test set)
Topics summarized:	24
Total shots returned:	685
Total relevant shots:	1428
Total relevant shots returned:	356
Precision at total relevant shots:	0.3277
Mean average precision:	0.2874

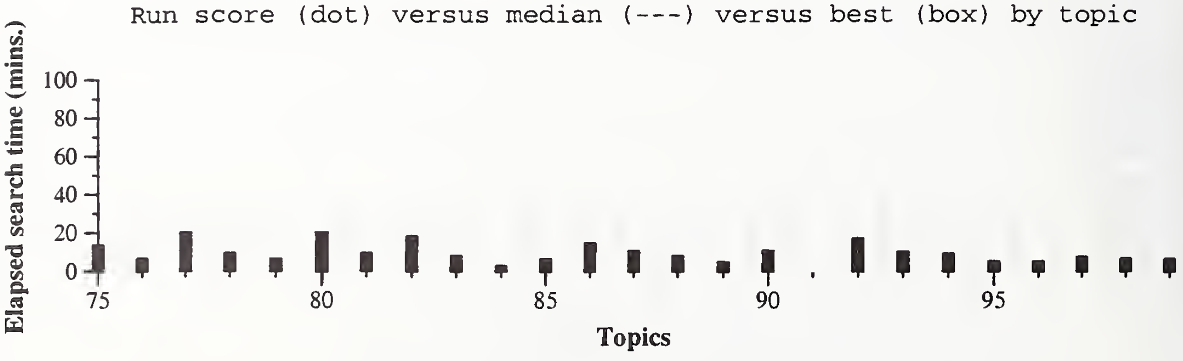


Interpolated recall	precision
0.0	0.9333
0.1	0.8495
0.2	0.6008
0.3	0.3821
0.4	0.2906
0.5	0.1976
0.6	0.1555
0.7	0.0717
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	0.7250
10	0.5958
15	0.5306
20	0.4688
30	0.3750
100	0.1483



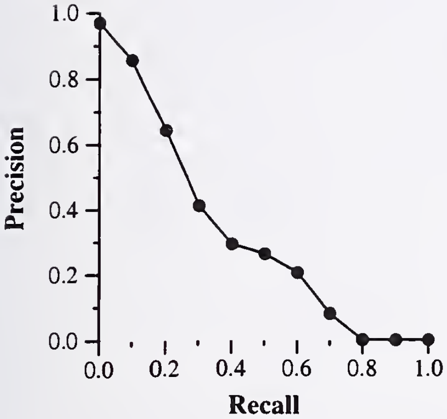
Run score (dot) versus median (---) versus best (box) by topic



Elapsed search time by topic

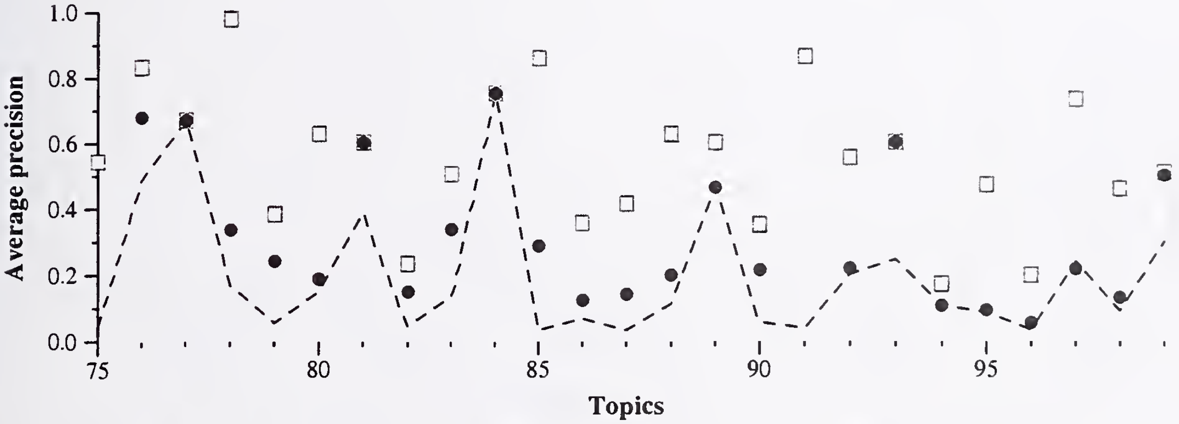
TREC 2002 Video Track: search results

Run ID:	DCUTrec11B.1
Processing type:	interactive
System training type:	B (no knowledge of test set)
Topics summarized:	23
Total shots returned:	549
Total relevant shots:	1413
Total relevant shots returned:	382
Precision at total relevant shots:	0.3536
Mean average precision:	0.3164

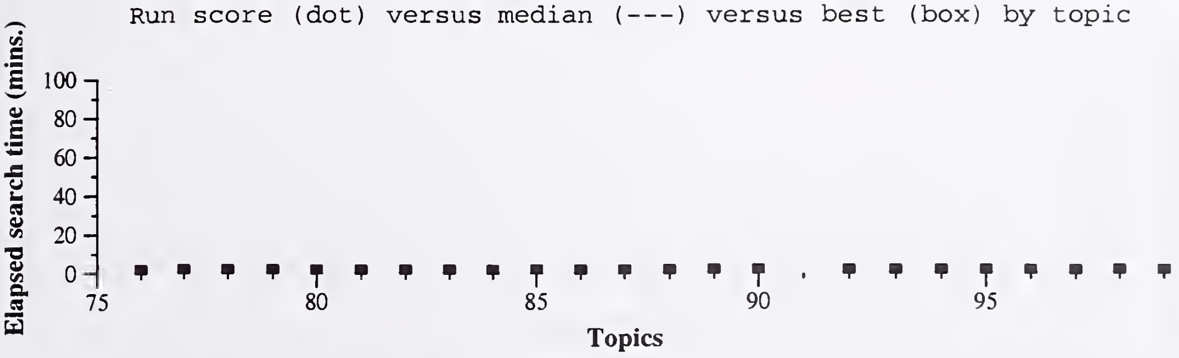


Interpolated recall	precision
0.0	0.9701
0.1	0.8564
0.2	0.6405
0.3	0.4085
0.4	0.2912
0.5	0.2611
0.6	0.2033
0.7	0.0793
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.7565
10	0.6435
15	0.5797
20	0.5130
30	0.4087
100	0.1600



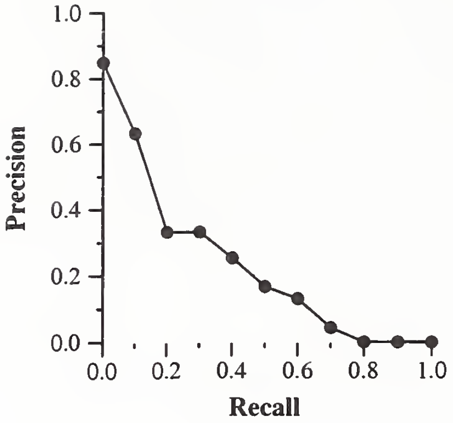
Run score (dot) versus median (---) versus best (box) by topic



Elapsed search time by topic

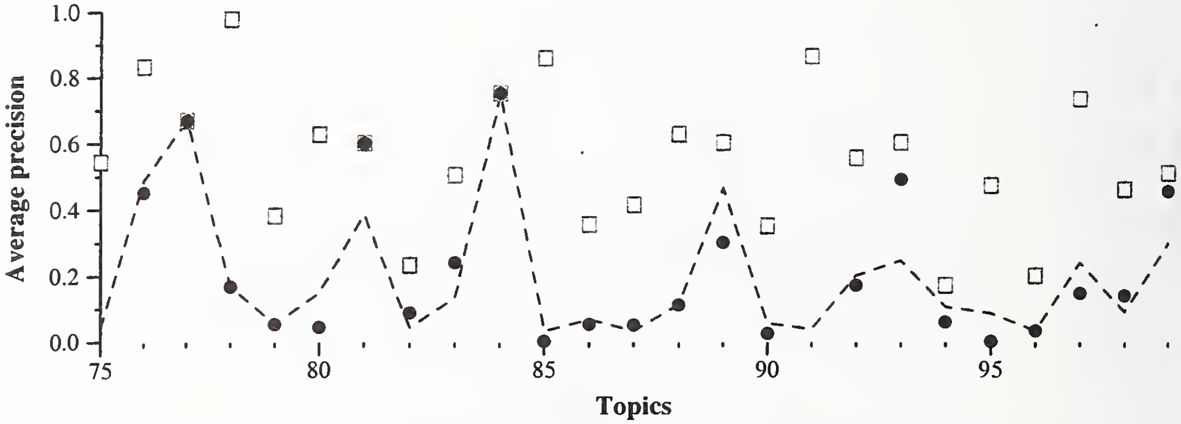
TREC 2002 Video Track: search results

Run ID:	DCUTrec11B.3
Processing type:	interactive
System training type:	B (no knowledge of test set)
Topics summarized:	23
Total shots returned:	287
Total relevant shots:	1413
Total relevant shots returned:	240
Precision at total relevant shots:	0.2390
Mean average precision:	0.2212

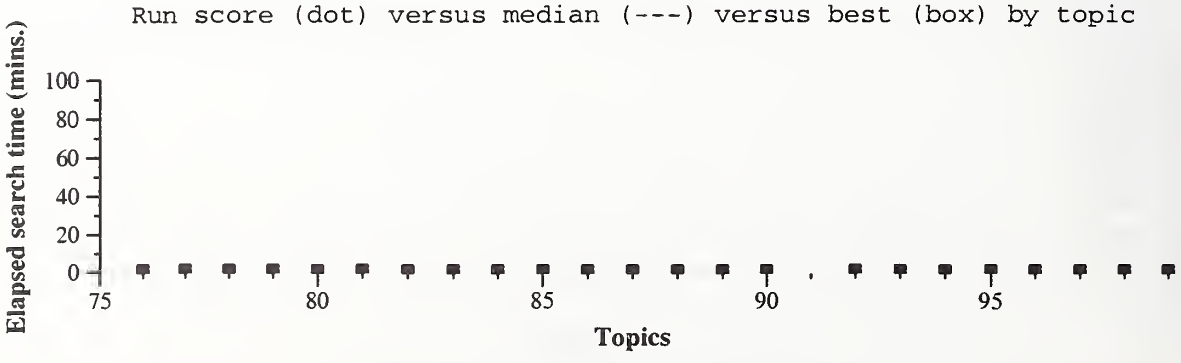


Interpolated recall	precision
0.0	0.8490
0.1	0.6321
0.2	0.3311
0.3	0.3311
0.4	0.2527
0.5	0.1667
0.6	0.1304
0.7	0.0435
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	0.6174
10	0.5087
15	0.4348
20	0.3674
30	0.2739
100	0.1043



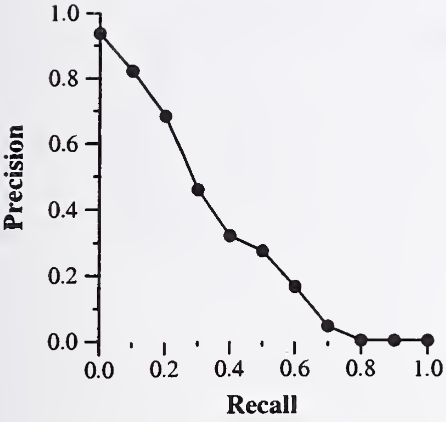
Run score (dot) versus median (---) versus best (box) by topic



Elapsed search time by topic

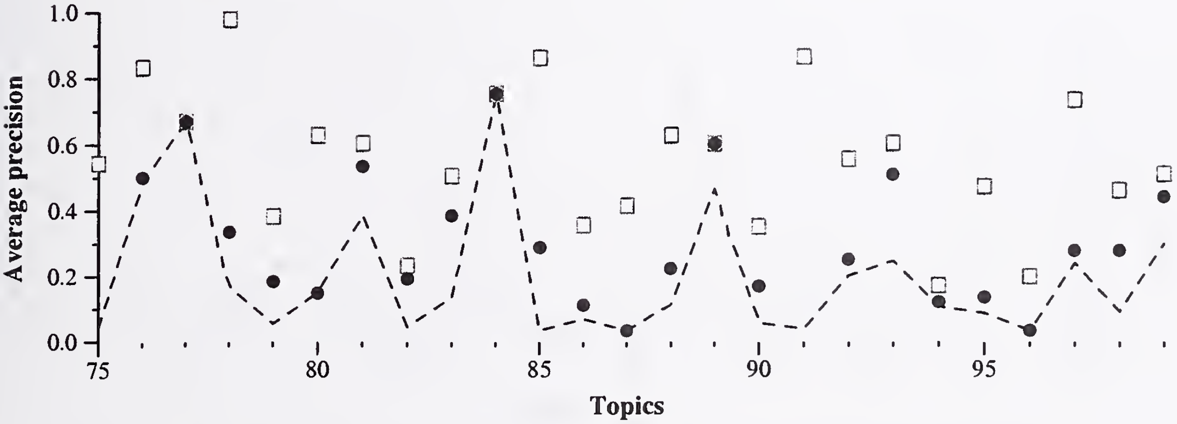
TREC 2002 Video Track: search results

Run ID:	DCUTrec11C.2
Processing type:	interactive
System training type:	B (no knowledge of test set)
Topics summarized:	23
Total shots returned:	595
Total relevant shots:	1413
Total relevant shots returned:	388
Precision at total relevant shots:	0.3547
Mean average precision:	0.3105

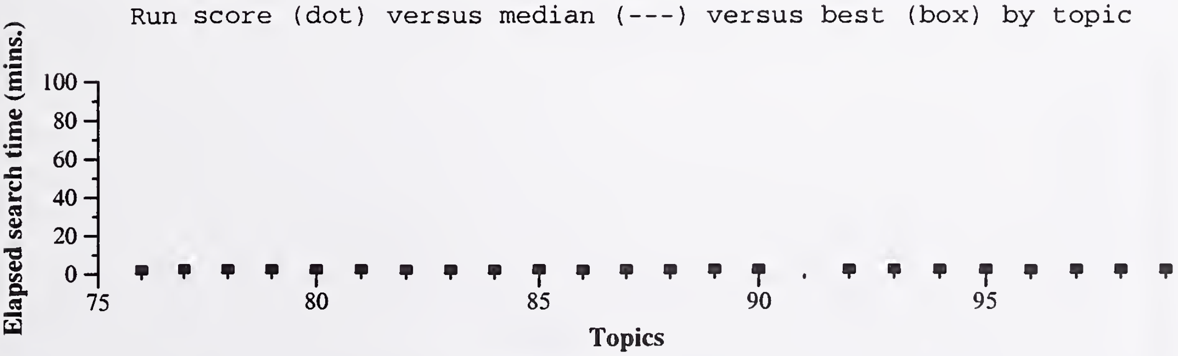


Interpolated recall	precision
0.0	0.9380
0.1	0.8216
0.2	0.6807
0.3	0.4556
0.4	0.3171
0.5	0.2712
0.6	0.1628
0.7	0.0435
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	0.7217
10	0.6174
15	0.5652
20	0.5152
30	0.4116
100	0.1678



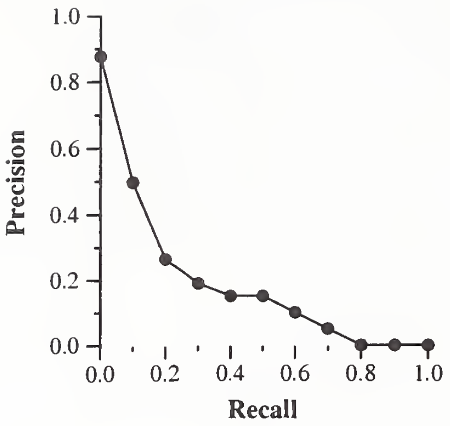
Run score (dot) versus median (---) versus best (box) by topic



Elapsed search time by topic

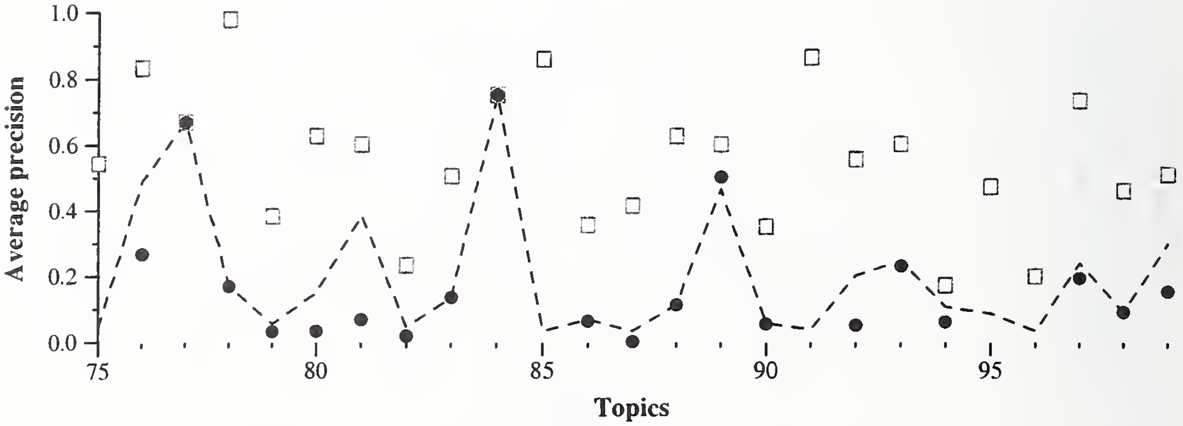
TREC 2002 Video Track: search results

Run ID:	DCUTrec11C.4
Processing type:	interactive
System training type:	B (no knowledge of test set)
Topics summarized:	20
Total shots returned:	236
Total relevant shots:	1358
Total relevant shots returned:	157
Precision at total relevant shots:	0.2031
Mean average precision:	0.1814

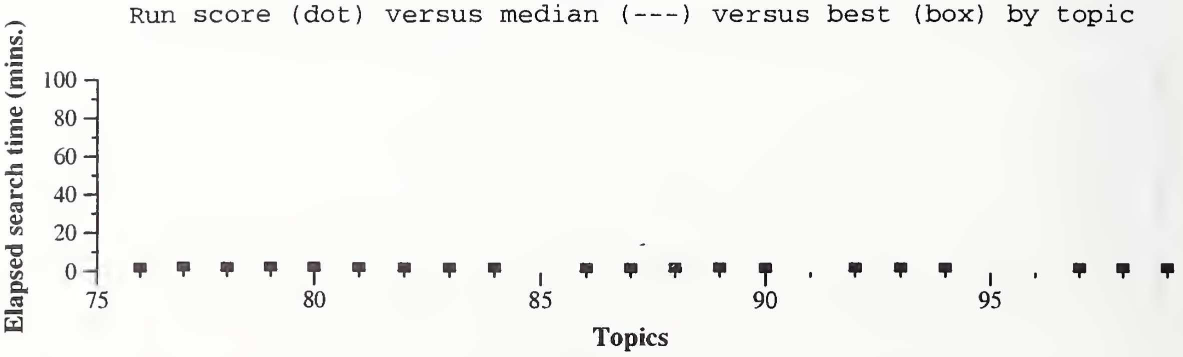


Interpolated recall	precision
0.0	0.8771
0.1	0.4971
0.2	0.2636
0.3	0.1881
0.4	0.1500
0.5	0.1500
0.6	0.1000
0.7	0.0500
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.5800
10	0.4450
15	0.3733
20	0.3175
30	0.2367
100	0.0785



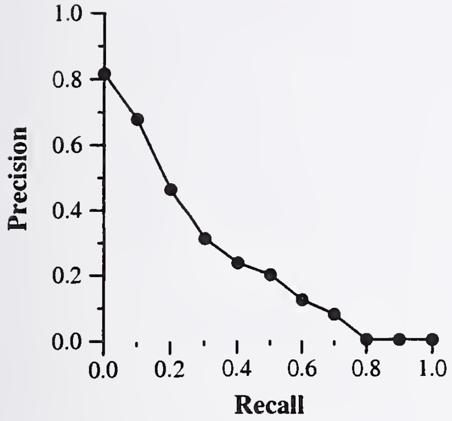
Run score (dot) versus median (---) versus best (box) by topic



Elapsed search time by topic

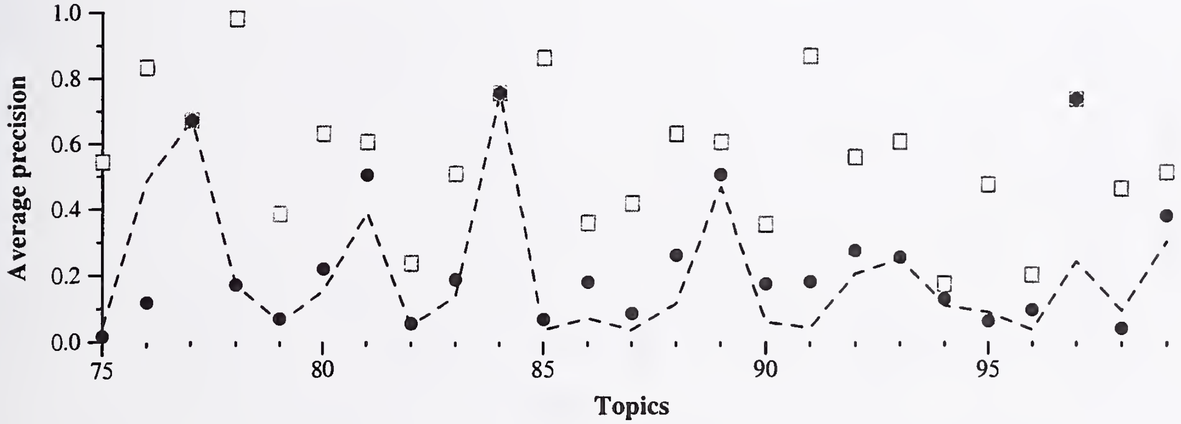
TREC 2002 Video Track: search results

Run ID:	IBM-4
Processing type:	interactive
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	2492
Total relevant shots:	1445
Total relevant shots returned:	398
Precision at total relevant shots:	0.2901
Mean average precision:	0.2441

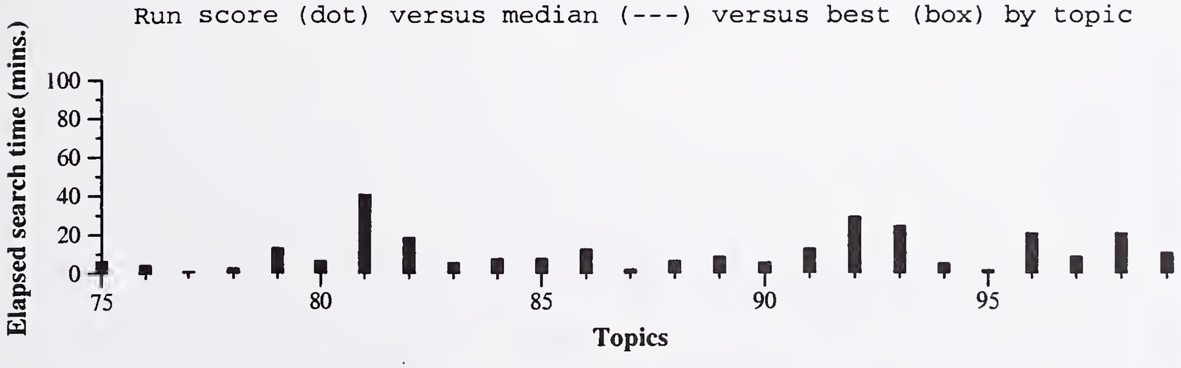


Interpolated recall	precision
0.0	0.8166
0.1	0.6772
0.2	0.4612
0.3	0.3073
0.4	0.2327
0.5	0.1960
0.6	0.1200
0.7	0.0763
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	0.5840
10	0.4800
15	0.4373
20	0.3880
30	0.3067
100	0.1592



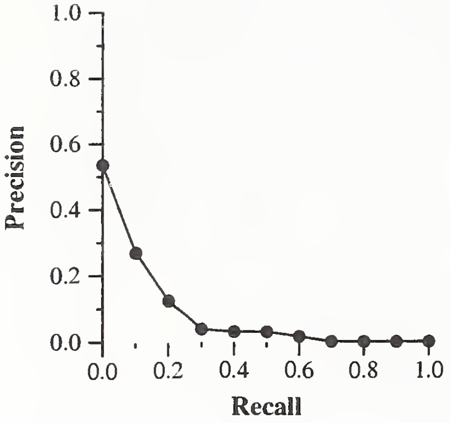
Run score (dot) versus median (---) versus best (box) by topic



Elapsed search time by topic

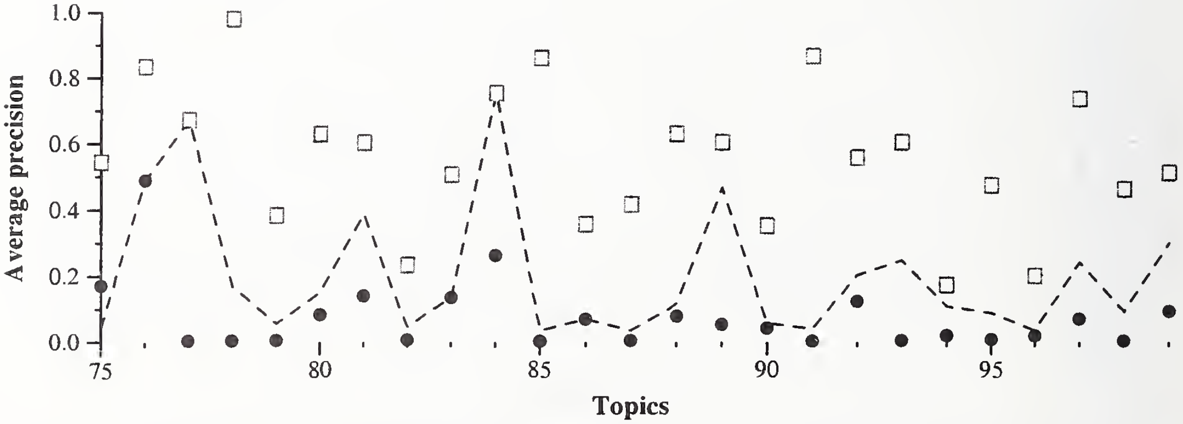
TREC 2002 Video Track: search results

Run ID:	ICMKM-1
Processing type:	interactive
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	2492
Total relevant shots:	1445
Total relevant shots returned:	176
Precision at total relevant shots:	0.1114
Mean average precision:	0.0735

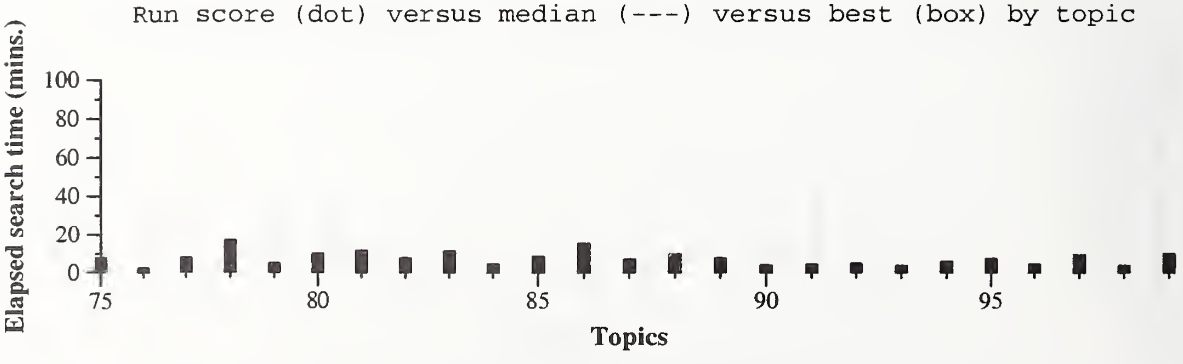


Interpolated recall	precision
0.0	0.5362
0.1	0.2681
0.2	0.1233
0.3	0.0365
0.4	0.0288
0.5	0.0283
0.6	0.0148
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.2400
10	0.2120
15	0.1813
20	0.1600
30	0.1293
100	0.0704



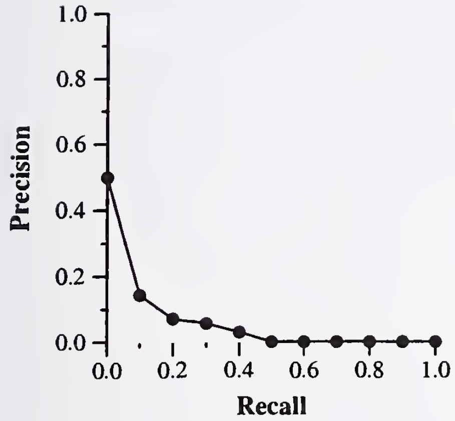
Run score (dot) versus median (---) versus best (box) by topic



Elapsed search time by topic

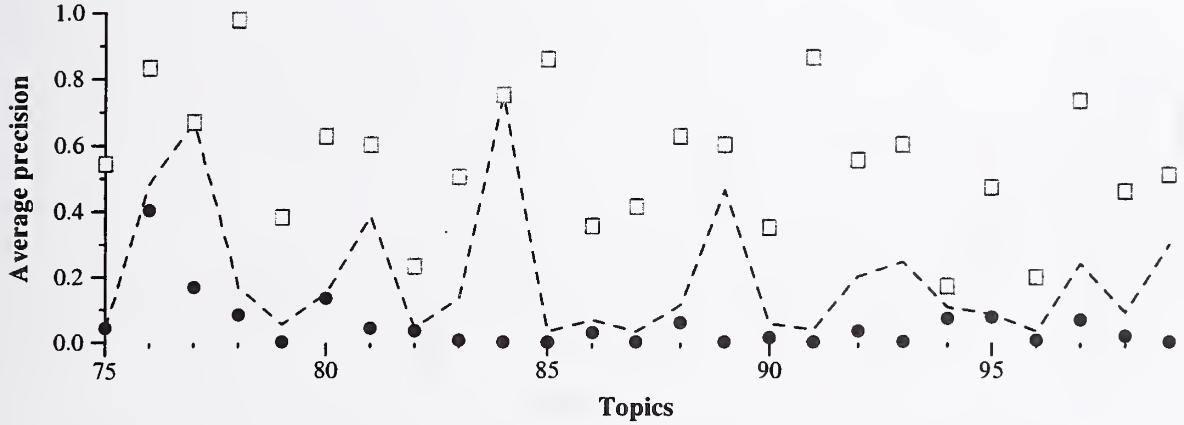
TREC 2002 Video Track: search results

Run ID:	MSRA.Q-Video.1
Processing type:	interactive
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	1243
Total relevant shots:	1445
Total relevant shots returned:	117
Precision at total relevant shots:	0.0891
Mean average precision:	0.0514

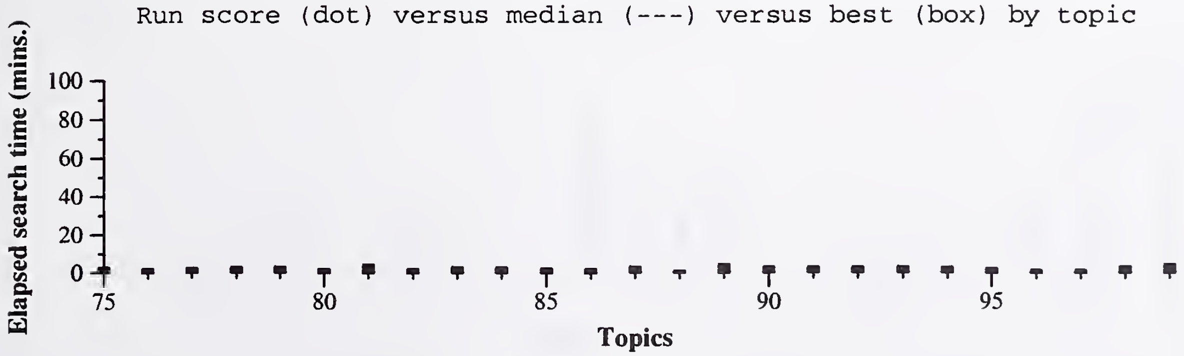


Interpolated recall	precision
0.0	0.4990
0.1	0.1428
0.2	0.0706
0.3	0.0553
0.4	0.0292
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.2880
10	0.2120
15	0.1733
20	0.1560
30	0.1240
100	0.0468



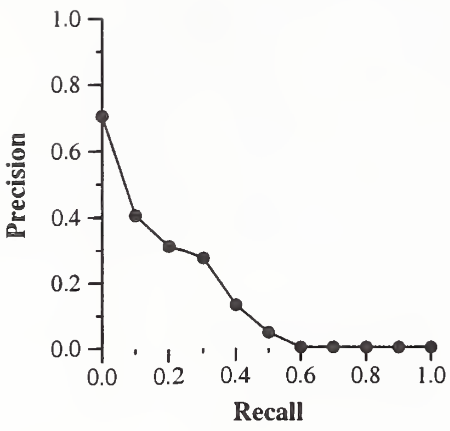
Run score (dot) versus median (---) versus best (box) by topic



Elapsed search time by topic

TREC 2002 Video Track: search results

Run ID:	MSRA.Q-Video.2.A
Processing type:	interactive
System training type:	B (no knowledge of test set)
Topics summarized:	19
Total shots returned:	945
Total relevant shots:	1144
Total relevant shots returned:	140
Precision at total relevant shots:	0.1916
Mean average precision:	0.1452

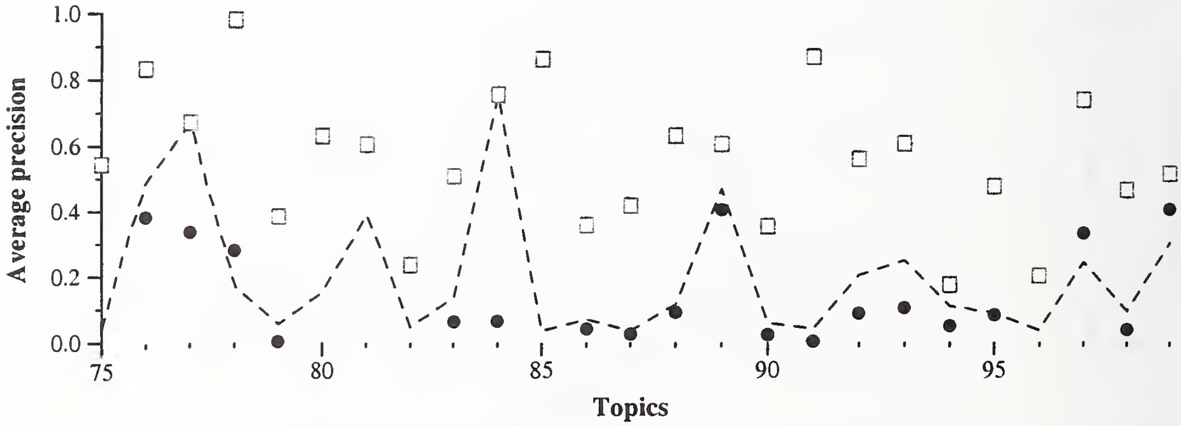


Interpolated recall precision

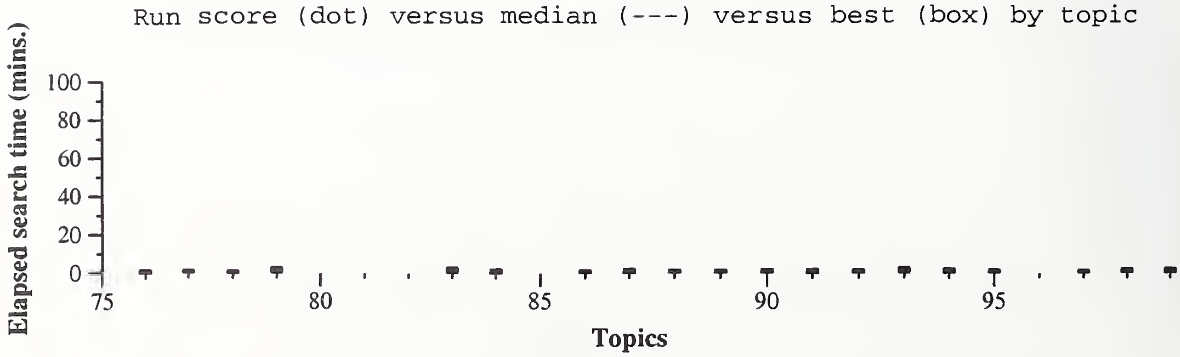
recall	precision
0.0	0.7058
0.1	0.4053
0.2	0.3109
0.3	0.2722
0.4	0.1290
0.5	0.0451
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots

n	shots
5	0.5158
10	0.4263
15	0.3474
20	0.2974
30	0.2246
100	0.0737



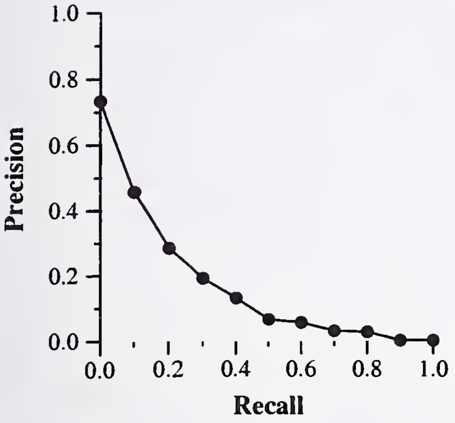
Run score (dot) versus median (---) versus best (box) by topic



Elapsed search time by topic

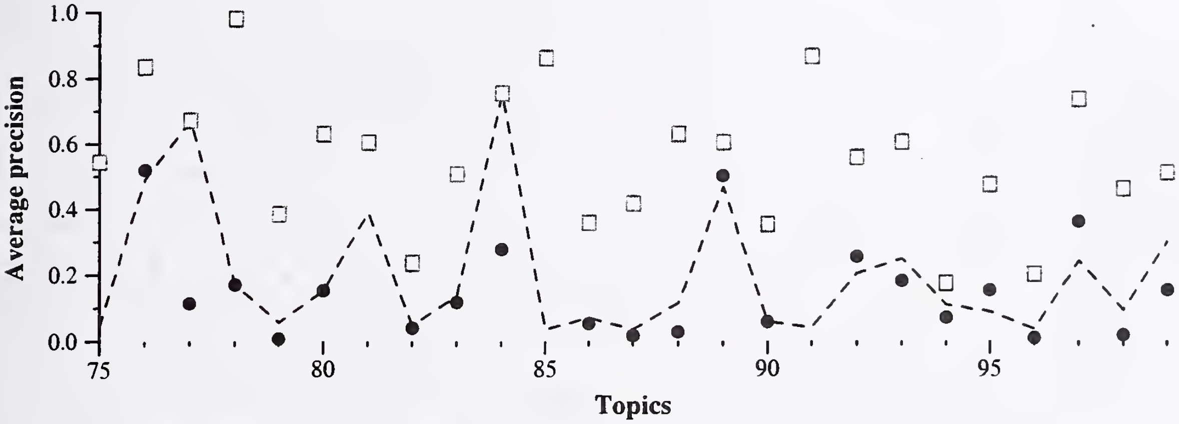
TREC 2002 Video Track: search results

Run ID:	UMDIqtrec
Processing type:	interactive
System training type:	B (no knowledge of test set)
Topics summarized:	21
Total shots returned:	1503
Total relevant shots:	1391
Total relevant shots returned:	293
Precision at total relevant shots:	0.2211
Mean average precision:	0.1519

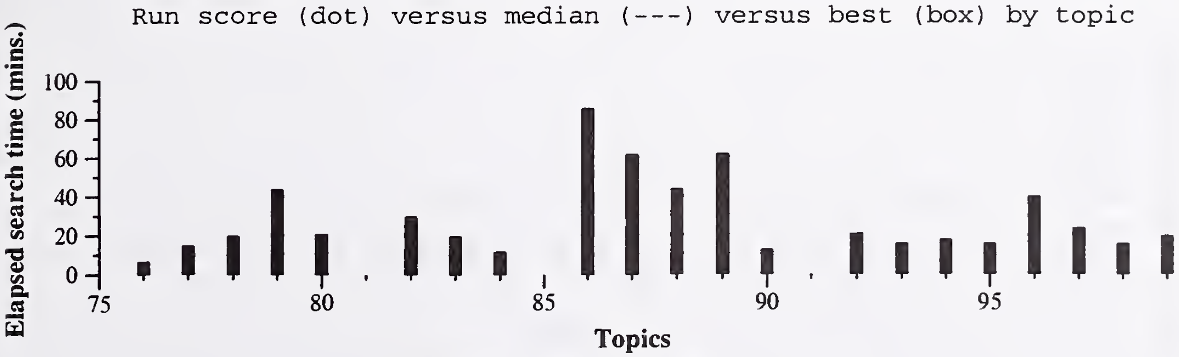


Interpolated recall	precision
0.0	0.7334
0.1	0.4559
0.2	0.2823
0.3	0.1886
0.4	0.1285
0.5	0.0636
0.6	0.0542
0.7	0.0282
0.8	0.0255
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	0.4190
10	0.3571
15	0.3175
20	0.2952
30	0.2524
100	0.1395



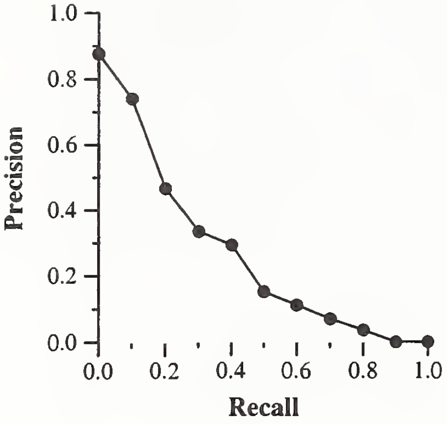
Run score (dot) versus median (---) versus best (box) by topic



Elapsed search time by topic

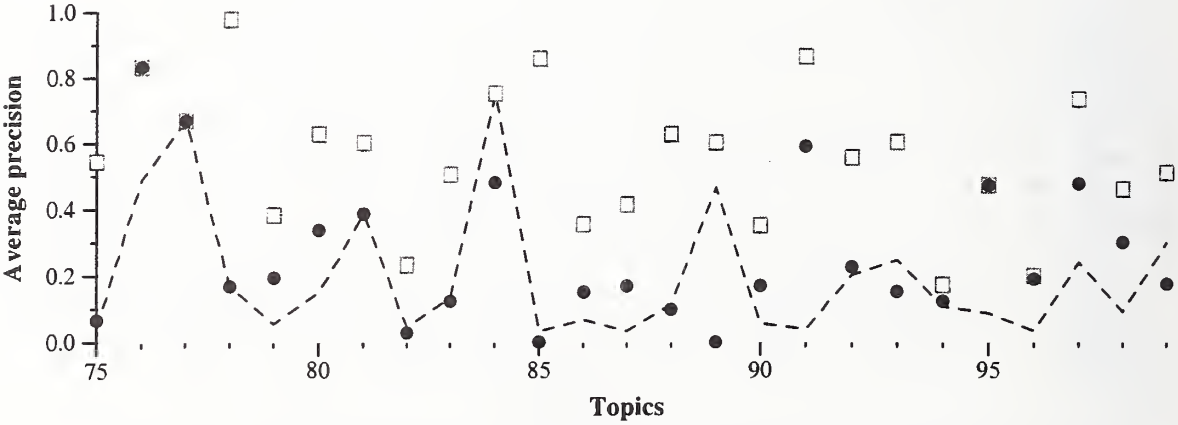
TREC 2002 Video Track: search results

Run ID:	UnivO_MT5
Processing type:	interactive
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	387
Total relevant shots:	1445
Total relevant shots returned:	317
Precision at total relevant shots:	0.2938
Mean average precision:	0.2622

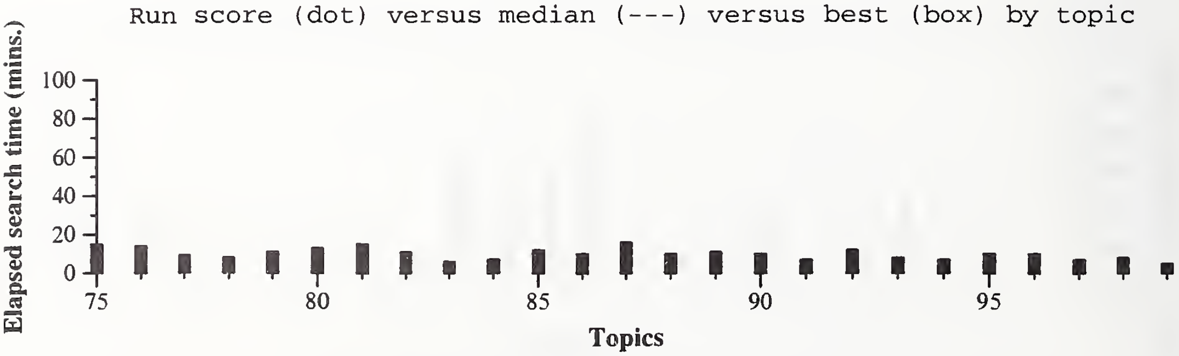


Interpolated recall	precision
0.0	0.8767
0.1	0.7401
0.2	0.4652
0.3	0.3325
0.4	0.2917
0.5	0.1500
0.6	0.1100
0.7	0.0688
0.8	0.0356
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	0.7360
10	0.6040
15	0.5227
20	0.4640
30	0.3693
100	0.1268



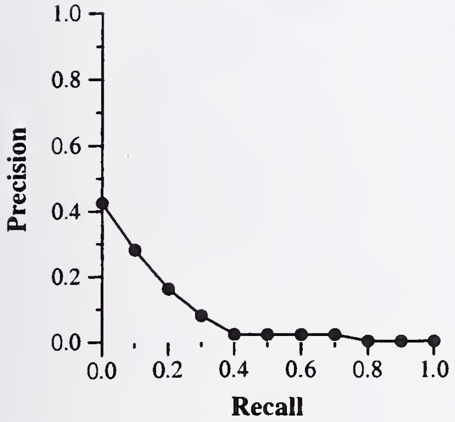
Run score (dot) versus median (---) versus best (box) by topic



Elapsed search time by topic

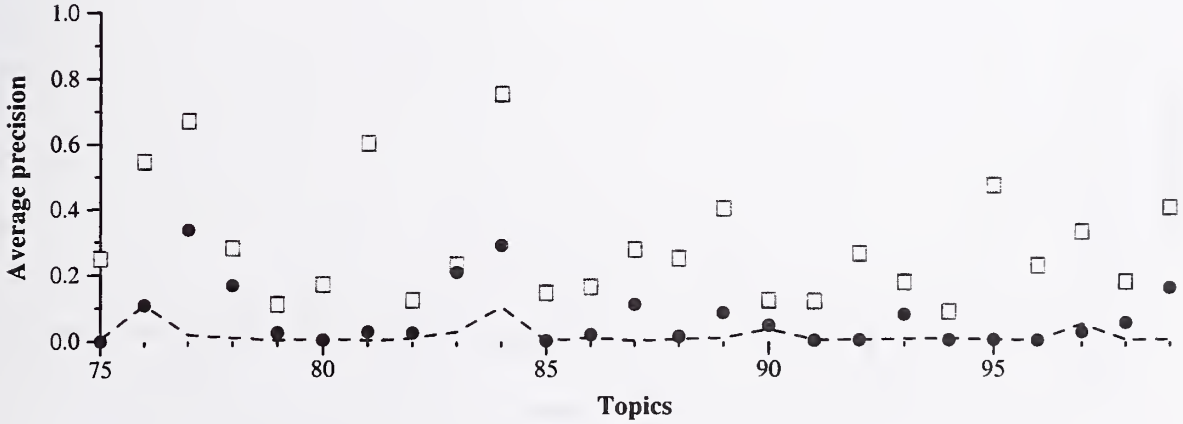
TREC 2002 Video Track: search results

Run ID:	CLIPS+ASR
Processing type:	manual
System training type:	A (with knowledge of test set)
Topics summarized:	25
Total shots returned:	2117
Total relevant shots:	1445
Total relevant shots returned:	181
Precision at total relevant shots:	0.1296
Mean average precision:	0.0708



Interpolated recall	precision
0.0	0.4255
0.1	0.2811
0.2	0.1608
0.3	0.0774
0.4	0.0200
0.5	0.0200
0.6	0.0200
0.7	0.0200
0.8	0.0000
0.9	0.0000
1.0	0.0000

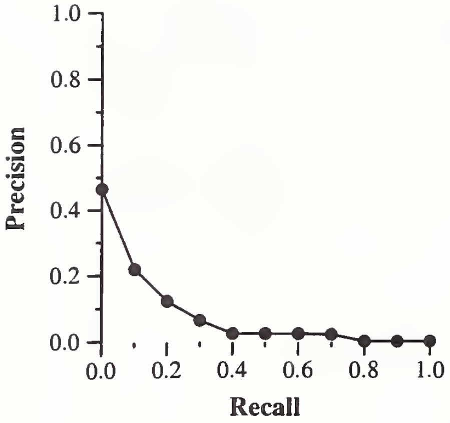
Precision at n	shots
5	0.1680
10	0.1560
15	0.1360
20	0.1280
30	0.1053
100	0.0724



Run score (dot) versus median (---) versus best (box) by topic

TREC 2002 Video Track: search results

Run ID:	CLIPS+ASR+X
Processing type:	manual
System training type:	A (with knowledge of test set)
Topics summarized:	25
Total shots returned:	2499
Total relevant shots:	1445
Total relevant shots returned:	96
Precision at total relevant shots:	0.0848
Mean average precision:	0.0636

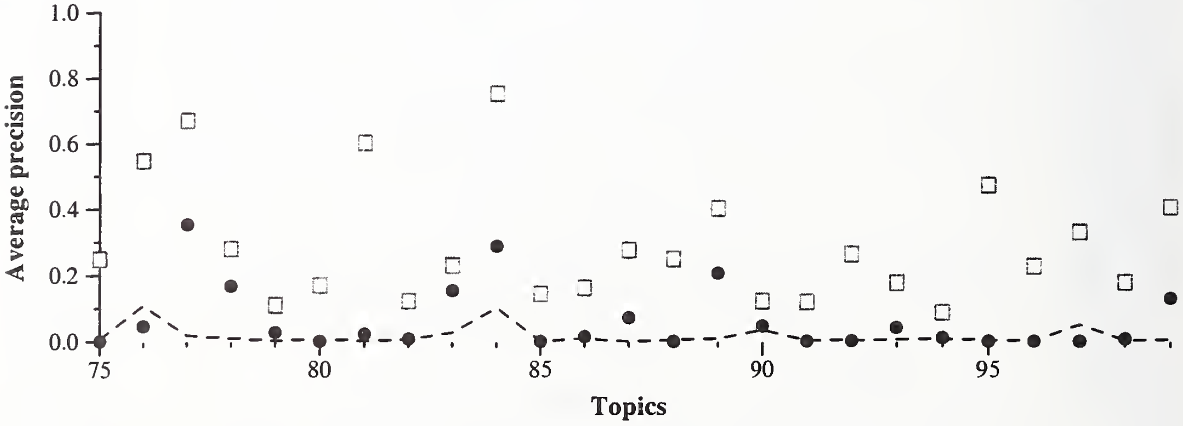


Interpolated recall precision

recall	precision
0.0	0.4646
0.1	0.2184
0.2	0.1218
0.3	0.0621
0.4	0.0222
0.5	0.0222
0.6	0.0222
0.7	0.0200
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots

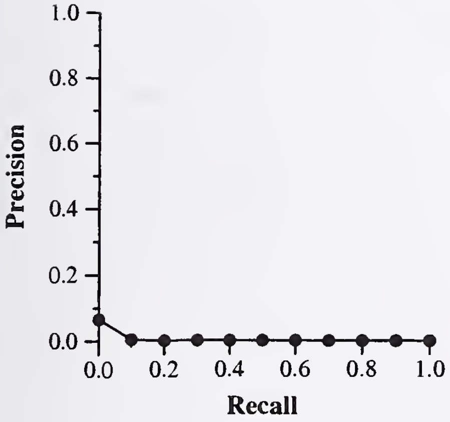
n	shots
5	0.2240
10	0.1520
15	0.1147
20	0.1020
30	0.0773
100	0.0384



Run score (dot) versus median (---) versus best (box) by topic

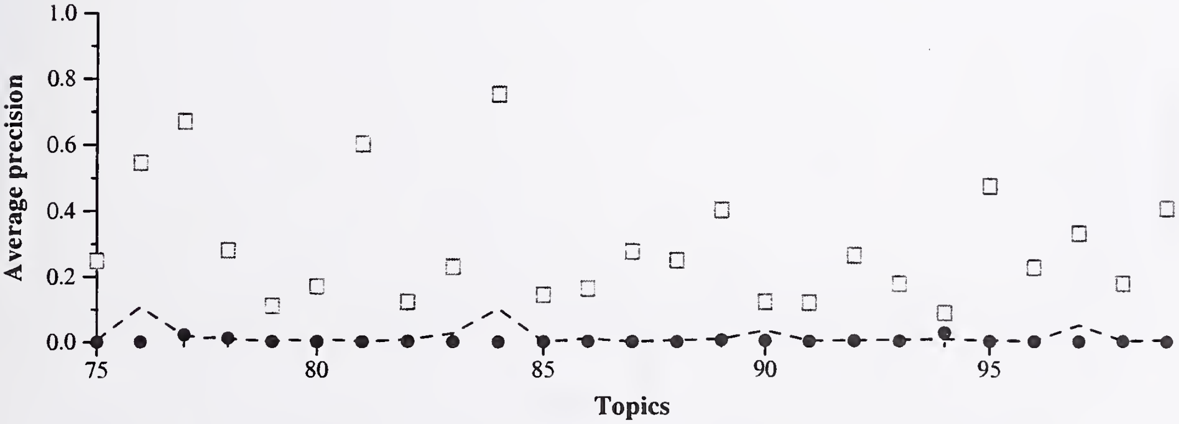
TREC 2002 Video Track: search results

Run ID:	CLIPS+X
Processing type:	manual
System training type:	A (with knowledge of test set)
Topics summarized:	25
Total shots returned:	2499
Total relevant shots:	1445
Total relevant shots returned:	40
Precision at total relevant shots:	0.0077
Mean average precision:	0.0029



Interpolated recall	precision
0.0	0.0661
0.1	0.0056
0.2	0.0012
0.3	0.0012
0.4	0.0011
0.5	0.0011
0.6	0.0011
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

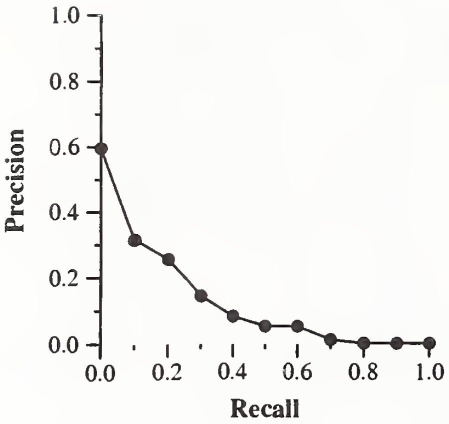
Precision at n shots	
5	0.0160
10	0.0240
15	0.0187
20	0.0280
30	0.0280
100	0.0160



Run score (dot) versus median (---) versus best (box) by topic

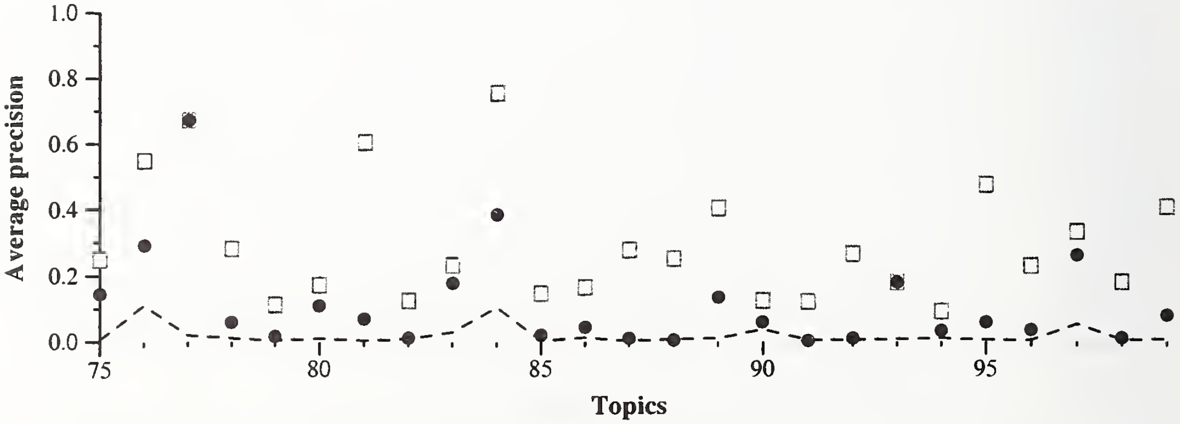
TREC 2002 Video Track: search results

Run ID:	CMU_MANUAL1
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	2500
Total relevant shots:	1445
Total relevant shots returned:	231
Precision at total relevant shots:	0.1783
Mean average precision:	0.1124



Interpolated recall	precision
0.0	0.5958
0.1	0.3125
0.2	0.2531
0.3	0.1421
0.4	0.0810
0.5	0.0509
0.6	0.0509
0.7	0.0109
0.8	0.0000
0.9	0.0000
1.0	0.0000

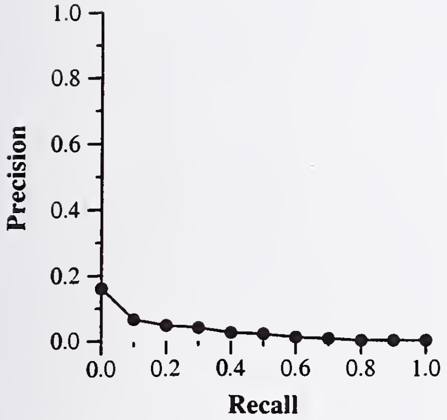
Precision at n shots	
5	0.2720
10	0.2440
15	0.2320
20	0.2080
30	0.1720
100	0.0924



Run score (dot) versus median (---) versus best (box) by topic

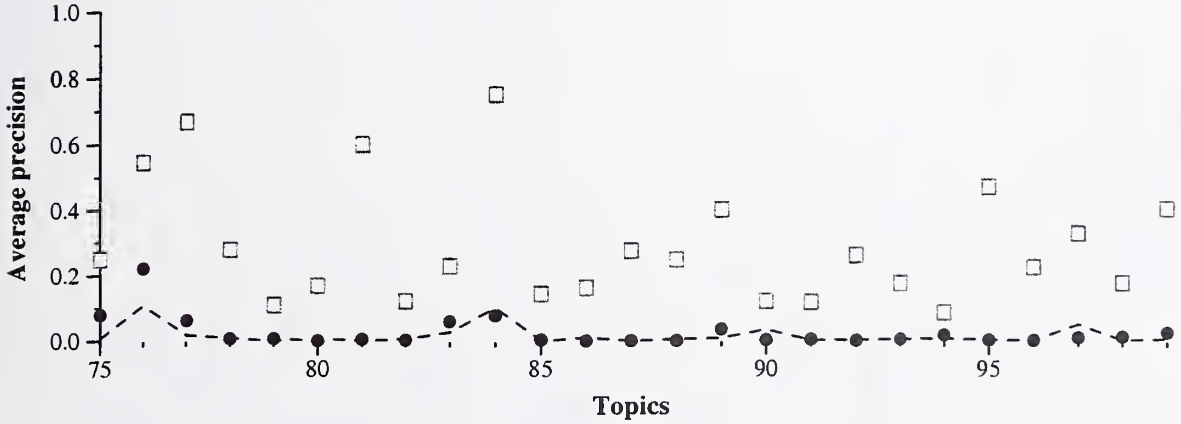
TREC 2002 Video Track: search results

Run ID:	CMU_MANUAL2
Processing type:	manual
System training type:	A (with knowledge of test set)
Topics summarized:	25
Total shots returned:	2500
Total relevant shots:	1445
Total relevant shots returned:	123
Precision at total relevant shots:	0.0448
Mean average precision:	0.0256



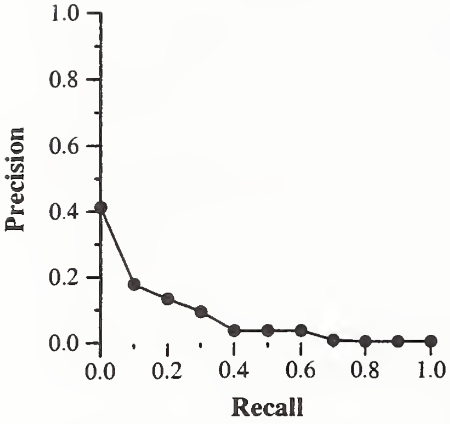
Interpolated recall	precision
0.0	0.1610
0.1	0.0650
0.2	0.0462
0.3	0.0379
0.4	0.0230
0.5	0.0193
0.6	0.0095
0.7	0.0055
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.0320
10	0.0320
15	0.0613
20	0.0700
30	0.0680
100	0.0492



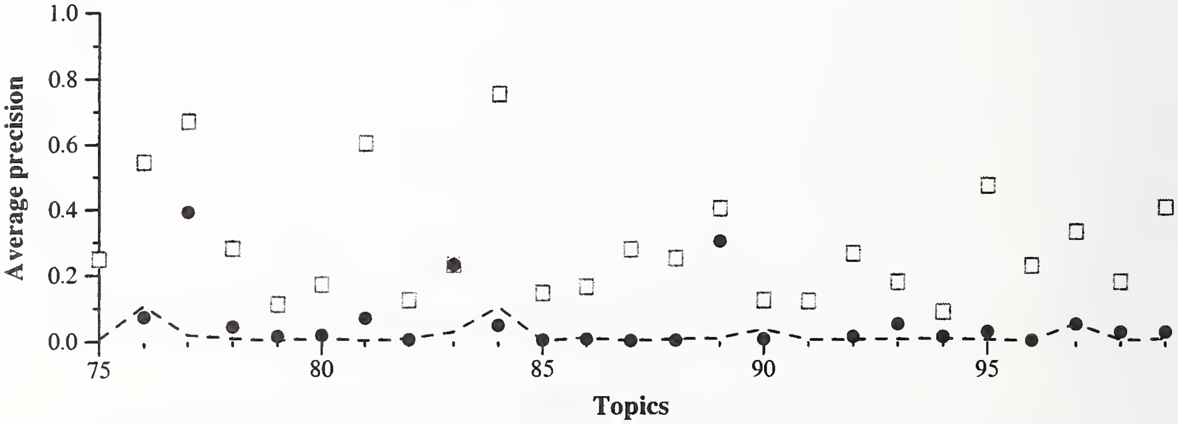
Run score (dot) versus median (---) versus best (box) by topic

```
TREC 2002 Video Track: search results

Run ID:                              CMU_MANUAL3
Processing type:                     manual
System training type:                A (with knowledge of test set)

Topics summarized:                   23

Total shots returned:                1666
Total relevant shots:                1413
Total relevant shots returned:       135

Precision at total relevant shots:   0.1066

Mean average precision:              0.0606
```



Interpolated recall	precision
0.0	0.4136
0.1	0.1781
0.2	0.1323
0.3	0.0906
0.4	0.0329
0.5	0.0329
0.6	0.0329
0.7	0.0040
0.8	0.0000
0.9	0.0000
1.0	0.0000

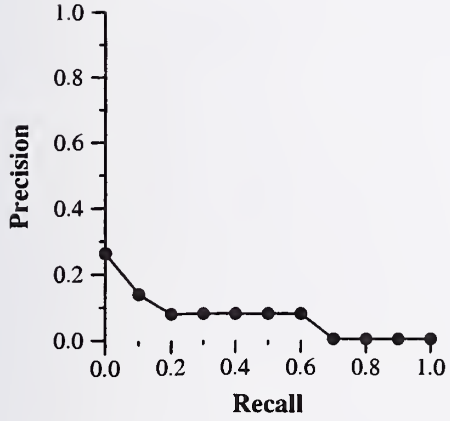
Precision at n	shots
5	0.1913
10	0.1261
15	0.1043
20	0.0870
30	0.0797
100	0.0587



Run score (dot) versus median (---) versus best (box) by topic

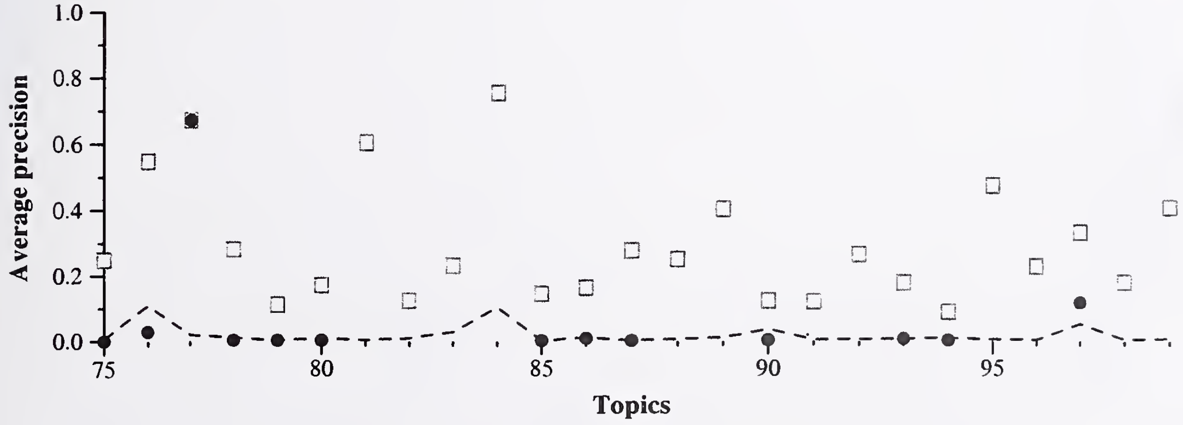
TREC 2002 Video Track: search results

Run ID:	Fudan_Search_Sys1
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	13
Total shots returned:	1201
Total relevant shots:	962
Total relevant shots returned:	33
Precision at total relevant shots:	0.0771
Mean average precision:	0.0632



Interpolated recall	precision
0.0	0.2640
0.1	0.1374
0.2	0.0769
0.3	0.0769
0.4	0.0769
0.5	0.0769
0.6	0.0769
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n	shots
5	0.1077
10	0.1000
15	0.0974
20	0.0769
30	0.0615
100	0.0254



Run score (dot) versus median (---) versus best (box) by topic

TREC 2002 Video Track: search results

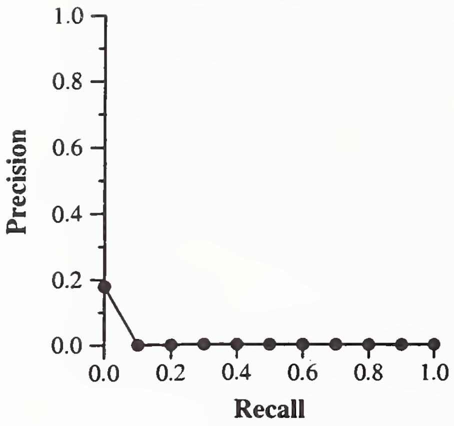
Run ID: Fudan_Search_Sys2
Processing type: manual
System training type: B (no knowledge of test set)

Topics summarized: 5

Total shots returned: 500
Total relevant shots: 383
Total relevant shots returned: 12

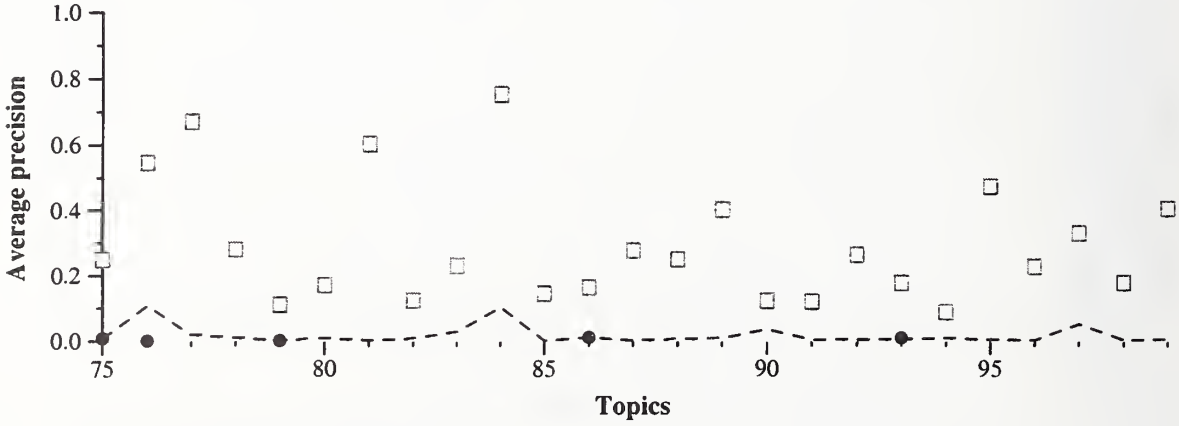
Precision at total relevant shots: 0.0310

Mean average precision: 0.0053



Interpolated recall	precision
0.0	0.1795
0.1	0.0000
0.2	0.0000
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

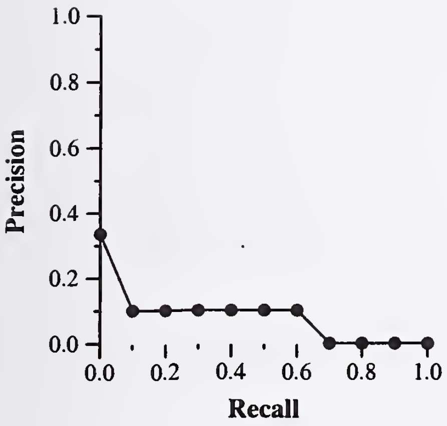
Precision at n shots	
5	0.0800
10	0.1000
15	0.0933
20	0.0700
30	0.0600
100	0.0240



Run score (dot) versus median (---) versus best (box) by topic

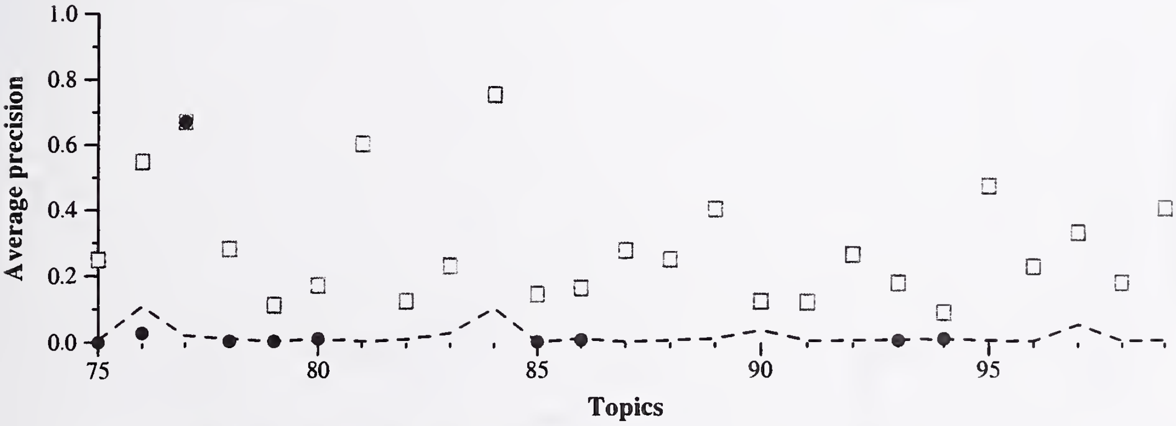
TREC 2002 Video Track: search results

Run ID:	Fudan_Search_Sys3
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	10
Total shots returned:	1000
Total relevant shots:	765
Total relevant shots returned:	27
Precision at total relevant shots:	0.0846
Mean average precision:	0.0717



Interpolated recall	precision
0.0	0.3362
0.1	0.1000
0.2	0.1000
0.3	0.1000
0.4	0.1000
0.5	0.1000
0.6	0.1000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

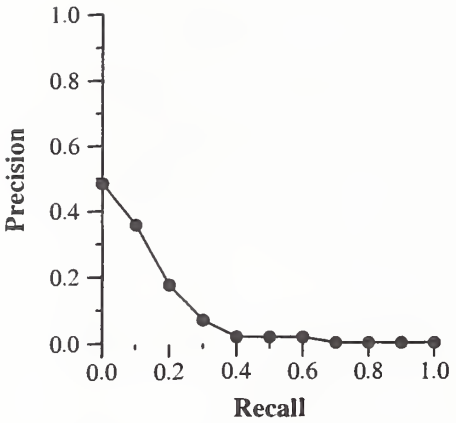
Precision at n shots	
5	0.1000
10	0.0700
15	0.0533
20	0.0550
30	0.0500
100	0.0270



Run score (dot) versus median (---) versus best (box) by topic

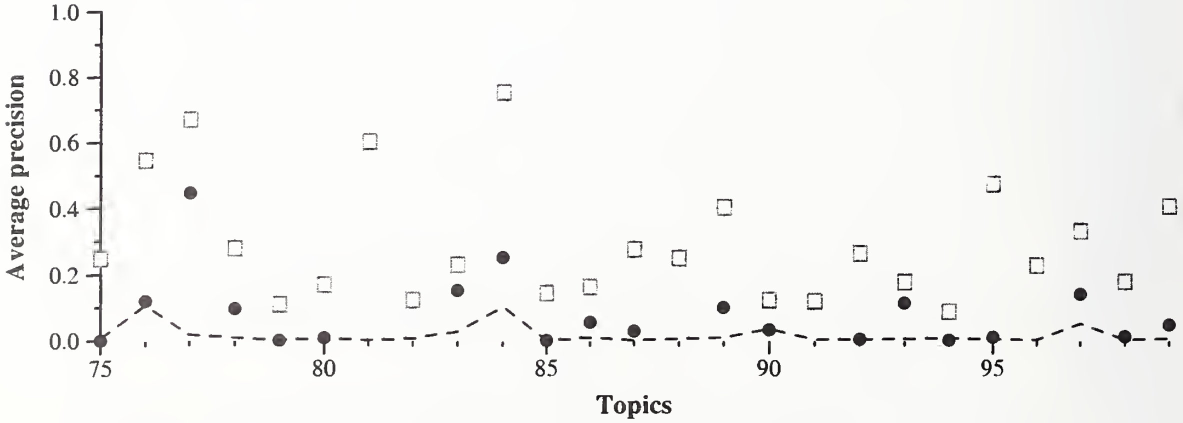
TREC 2002 Video Track: search results

Run ID:	Fudan_Search_Sys4
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	20
Total shots returned:	1353
Total relevant shots:	1140
Total relevant shots returned:	125
Precision at total relevant shots:	0.1254
Mean average precision:	0.0804



Interpolated recall	precision
0.0	0.4848
0.1	0.3570
0.2	0.1752
0.3	0.0667
0.4	0.0167
0.5	0.0167
0.6	0.0167
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

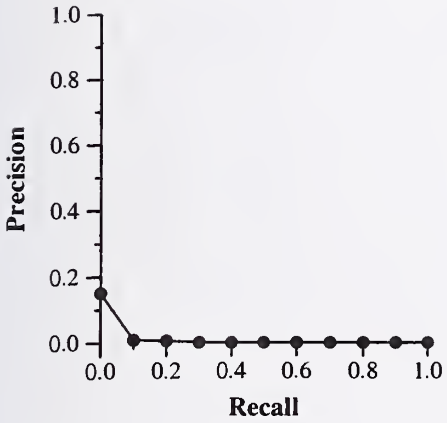
Precision at n shots	
5	0.2400
10	0.2150
15	0.1800
20	0.1600
30	0.1217
100	0.0625



Run score (dot) versus median (---) versus best (box) by topic

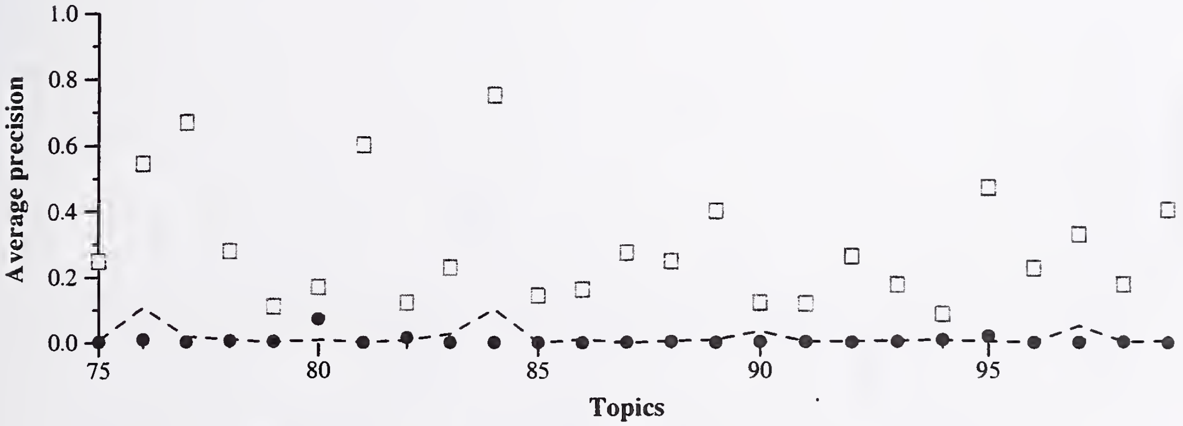
TREC 2002 Video Track: search results

Run ID:	IBM-1
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	2498
Total relevant shots:	1445
Total relevant shots returned:	64
Precision at total relevant shots:	0.0236
Mean average precision:	0.0059



Interpolated recall	precision
0.0	0.1519
0.1	0.0094
0.2	0.0067
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

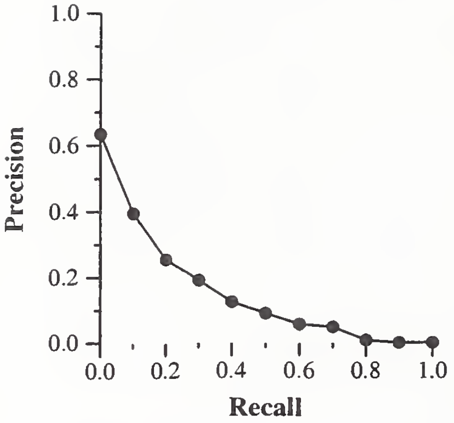
Precision at n	shots
5	0.0640
10	0.0400
15	0.0347
20	0.0340
30	0.0347
100	0.0256



Run score (dot) versus median (---) versus best (box) by topic

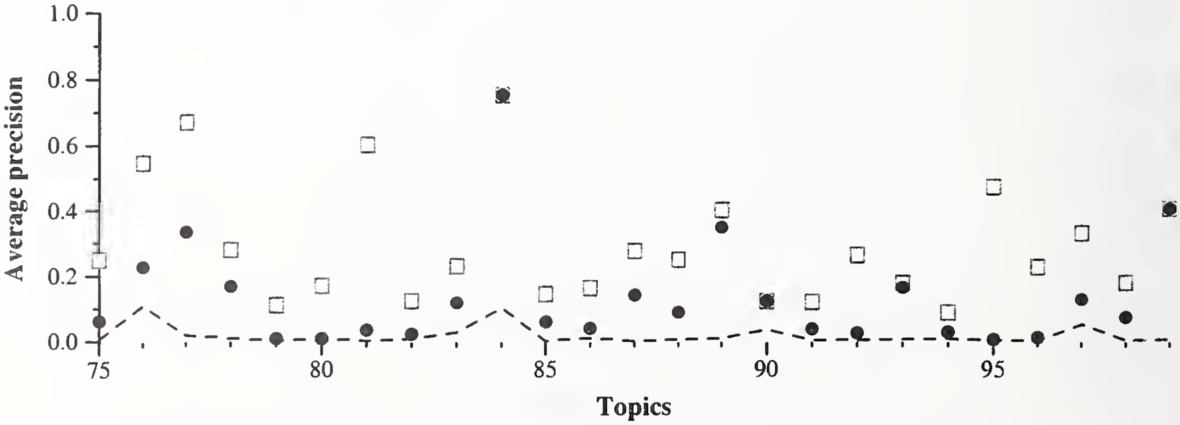
TREC 2002 Video Track: search results

Run ID:	IBM-2
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	2500
Total relevant shots:	1445
Total relevant shots returned:	256
Precision at total relevant shots:	0.2053
Mean average precision:	0.1357



Interpolated recall	precision
0.0	0.6347
0.1	0.3938
0.2	0.2530
0.3	0.1890
0.4	0.1237
0.5	0.0886
0.6	0.0556
0.7	0.0471
0.8	0.0071
0.9	0.0000
1.0	0.0000

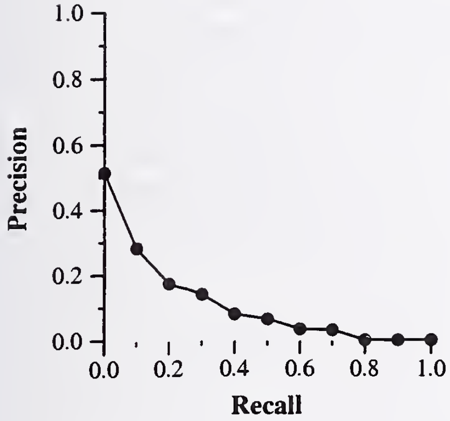
Precision at n shots	
5	0.3360
10	0.2720
15	0.2587
20	0.2440
30	0.2067
100	0.1024



Run score (dot) versus median (---) versus best (box) by topic

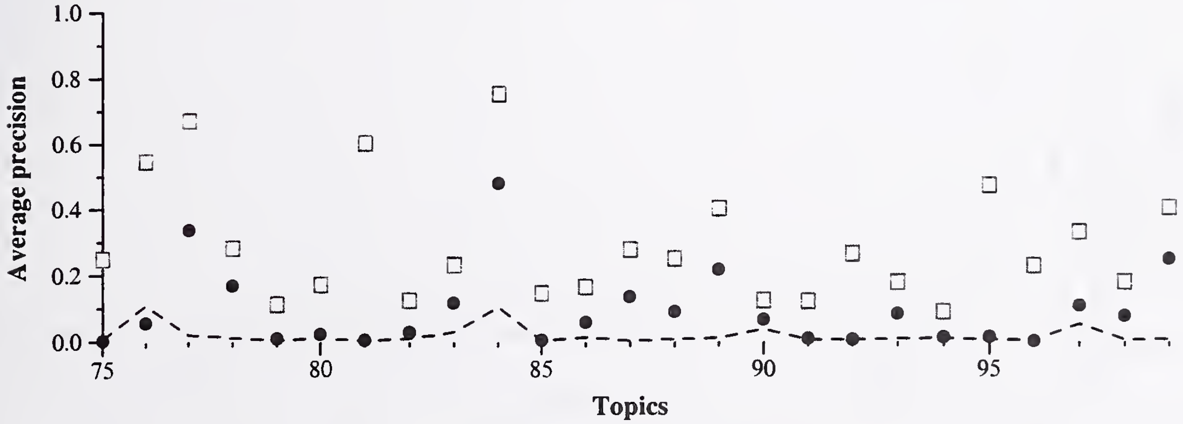
TREC 2002 Video Track: search results

Run ID:	IBM-3
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	2497
Total relevant shots:	1445
Total relevant shots returned:	229
Precision at total relevant shots:	0.1602
Mean average precision:	0.0926



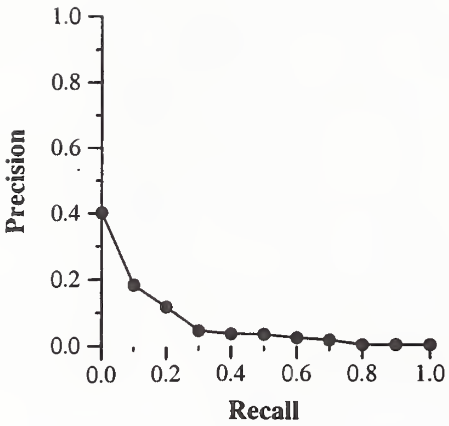
Interpolated recall	precision
0.0	0.5150
0.1	0.2819
0.2	0.1730
0.3	0.1387
0.4	0.0783
0.5	0.0629
0.6	0.0324
0.7	0.0300
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.3120
10	0.2240
15	0.1787
20	0.1520
30	0.1253
100	0.0916



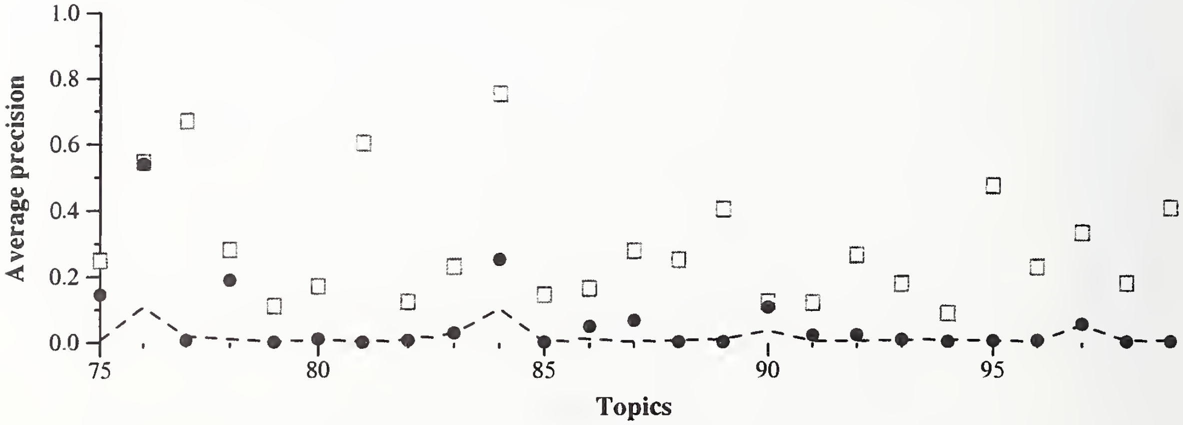
Run score (dot) versus median (---) versus best (box) by topic

```
TREC 2002 Video Track: search results

Run ID:                              ICMKM-2
Processing type:                     manual
System training type:                B (no knowledge of test set)

Topics summarized:                   25

Total shots returned:                2491
Total relevant shots:                1445
Total relevant shots returned:       138

Precision at total relevant shots:   0.0936

Mean average precision:              0.0600
```



Interpolated recall	precision
0.0	0.4021
0.1	0.1822
0.2	0.1160
0.3	0.0420
0.4	0.0326
0.5	0.0312
0.6	0.0211
0.7	0.0146
0.8	0.0000
0.9	0.0000
1.0	0.0000

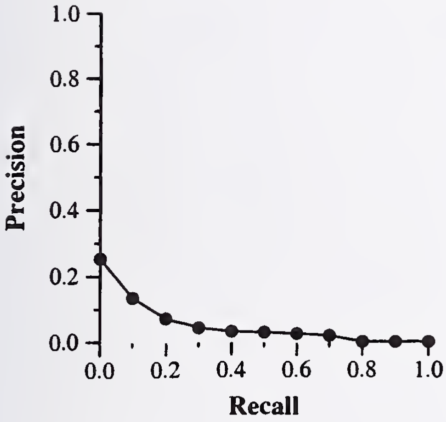
Precision at n shots	
5	0.1680
10	0.1280
15	0.1147
20	0.1140
30	0.1040
100	0.0552



Run score (dot) versus median (---) versus best (box) by topic

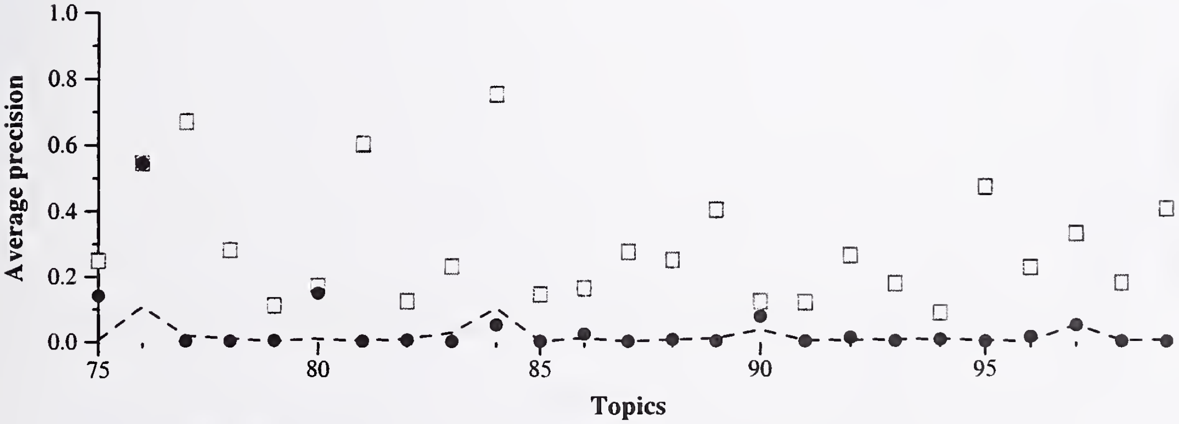
TREC 2002 Video Track: search results

Run ID:	ICMKM-3
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	2492
Total relevant shots:	1445
Total relevant shots returned:	133
Precision at total relevant shots:	0.0693
Mean average precision:	0.0430



Interpolated recall	precision
0.0	0.2535
0.1	0.1342
0.2	0.0711
0.3	0.0413
0.4	0.0307
0.5	0.0282
0.6	0.0237
0.7	0.0181
0.8	0.0000
0.9	0.0000
1.0	0.0000

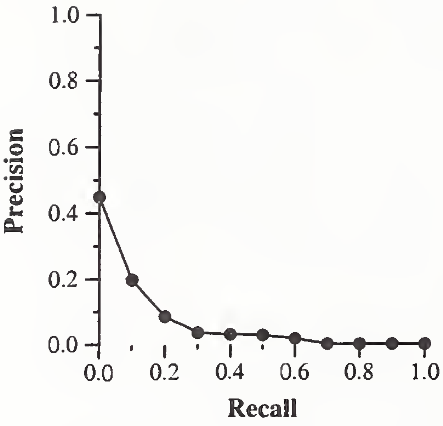
Precision at n	shots
5	0.1520
10	0.1160
15	0.1067
20	0.1080
30	0.0960
100	0.0532



Run score (dot) versus median (---) versus best (box) by topic

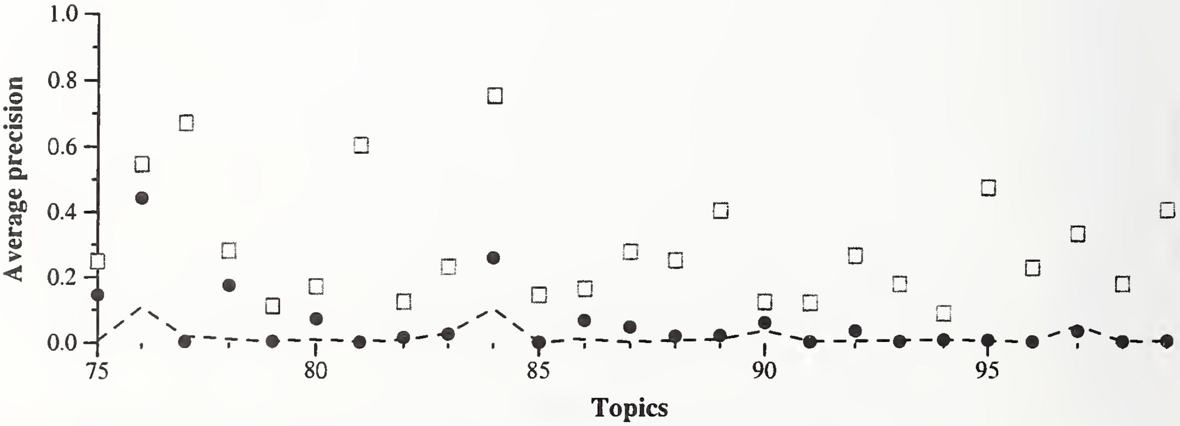
TREC 2002 Video Track: search results

Run ID:	ICMKM-4
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	2491
Total relevant shots:	1445
Total relevant shots returned:	132
Precision at total relevant shots:	0.0859
Mean average precision:	0.0568



Interpolated recall	precision
0.0	0.4487
0.1	0.1966
0.2	0.0837
0.3	0.0324
0.4	0.0279
0.5	0.0258
0.6	0.0157
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

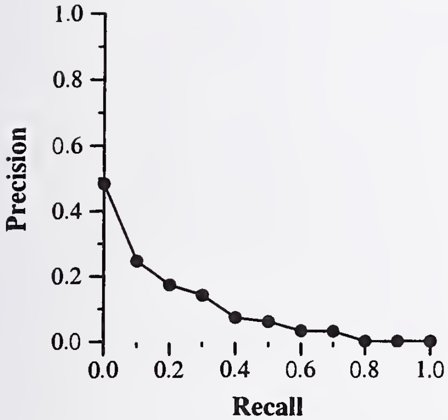
Precision at n shots	
5	0.2080
10	0.1720
15	0.1547
20	0.1380
30	0.1120
100	0.0528



Run score (dot) versus median (---) versus best (box) by topic

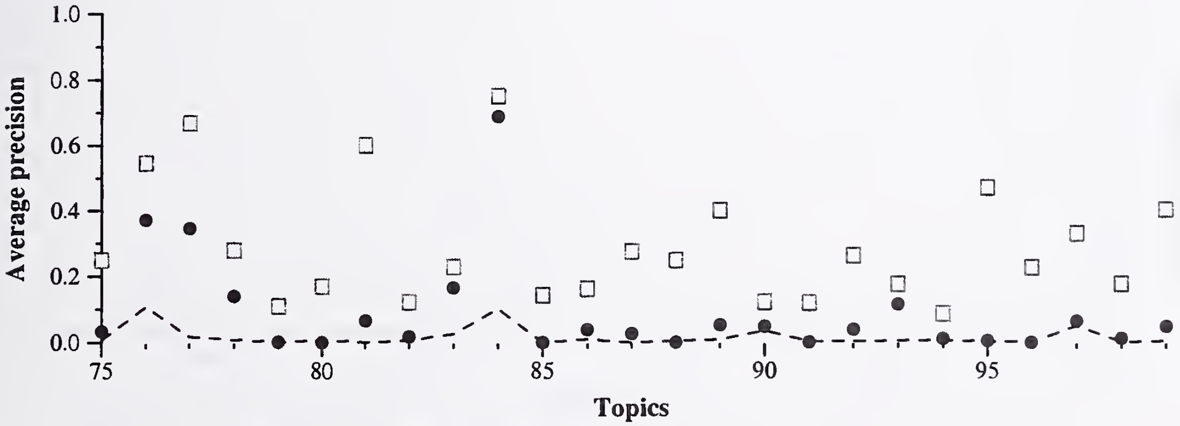
TREC 2002 Video Track: search results

Run ID:	LL10_T
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	2456
Total relevant shots:	1445
Total relevant shots returned:	181
Precision at total relevant shots:	0.1410
Mean average precision:	0.0917



Interpolated recall	precision
0.0	0.4843
0.1	0.2469
0.2	0.1732
0.3	0.1401
0.4	0.0718
0.5	0.0594
0.6	0.0313
0.7	0.0300
0.8	0.0000
0.9	0.0000
1.0	0.0000

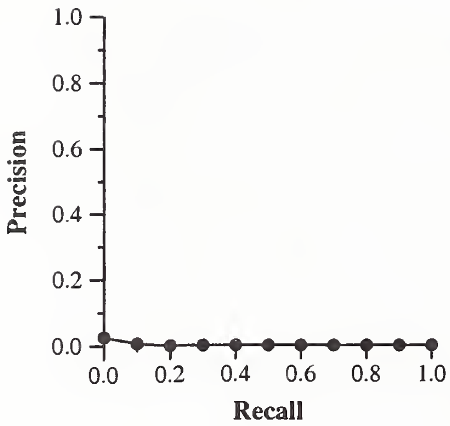
Precision at n shots	
5	0.2400
10	0.1920
15	0.1680
20	0.1460
30	0.1280
100	0.0724



Run score (dot) versus median (---) versus best (box) by topic

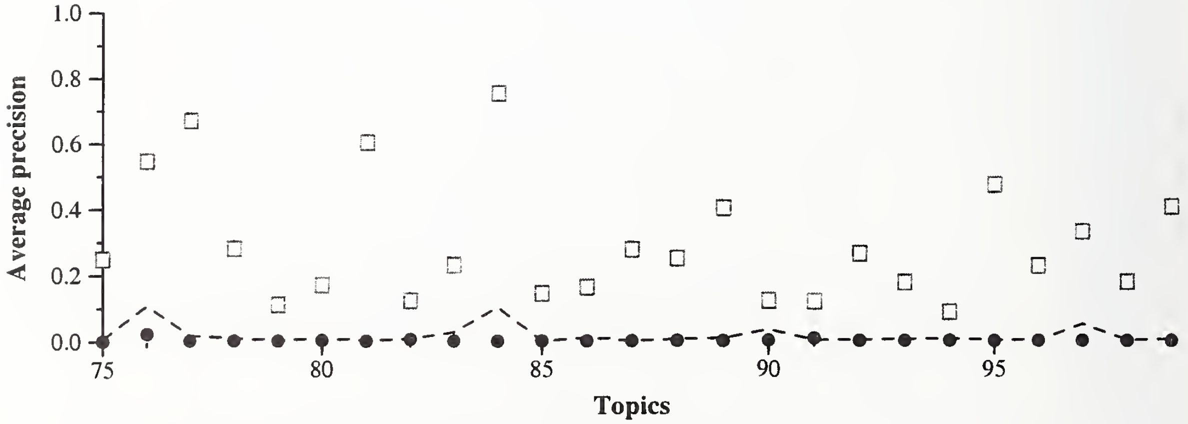
TREC 2002 Video Track: search results

Run ID:	LL10_TIac
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	2497
Total relevant shots:	1445
Total relevant shots returned:	30
Precision at total relevant shots:	0.0093
Mean average precision:	0.0016



Interpolated recall	precision
0.0	0.0262
0.1	0.0069
0.2	0.0000
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

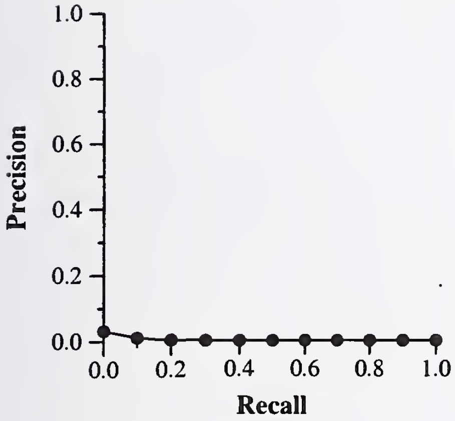
Precision at n shots	
5	0.0080
10	0.0040
15	0.0080
20	0.0120
30	0.0147
100	0.0120



Run score (dot) versus median (---) versus best (box) by topic

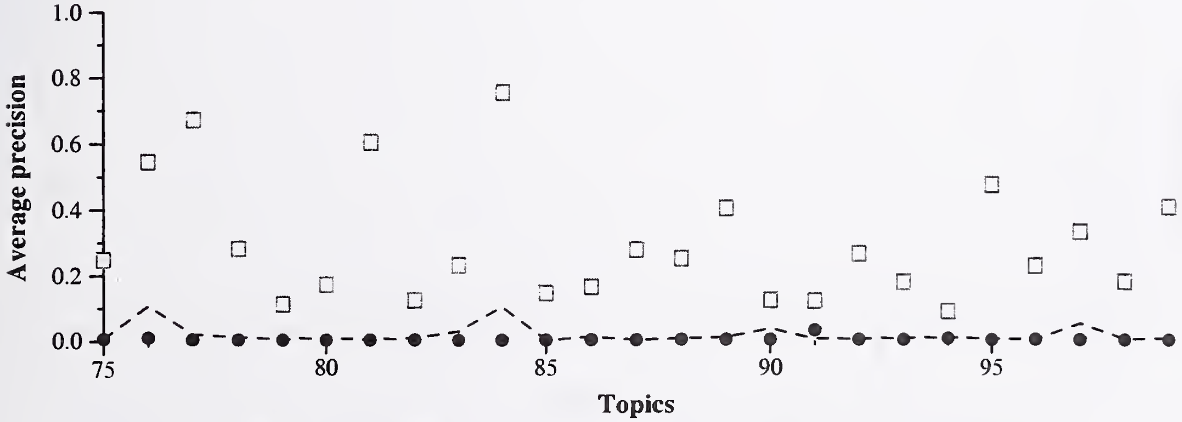
TREC 2002 Video Track: search results

Run ID:	LL10_TIsc
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	2499
Total relevant shots:	1445
Total relevant shots returned:	35
Precision at total relevant shots:	0.0167
Mean average precision:	0.0022



Interpolated recall	precision
0.0	0.0319
0.1	0.0115
0.2	0.0037
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

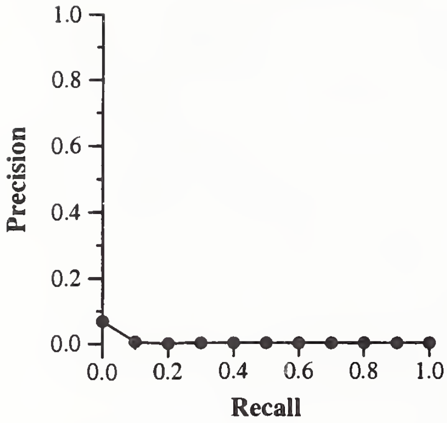
Precision at n	shots
5	0.0000
10	0.0080
15	0.0160
20	0.0140
30	0.0147
100	0.0140



Run score (dot) versus median (---) versus best (box) by topic

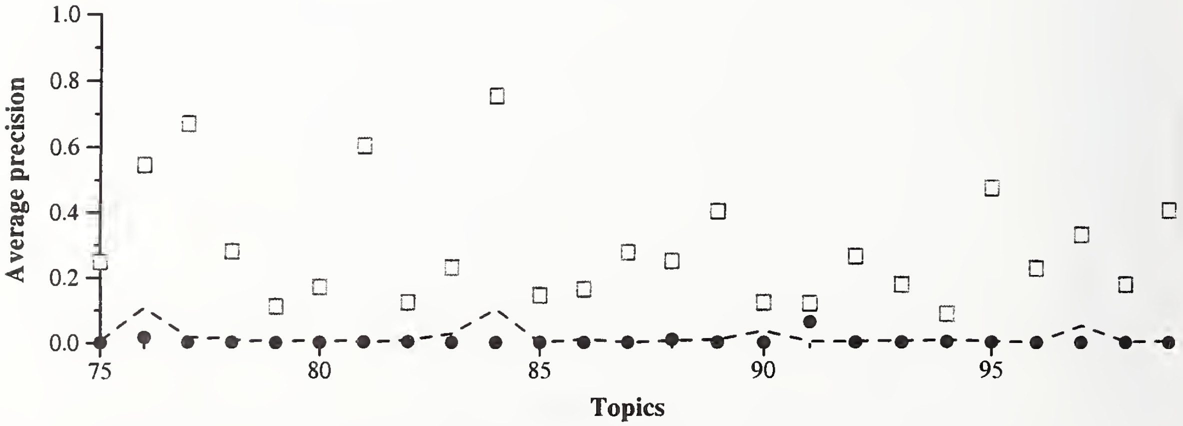
TREC 2002 Video Track: search results

Run ID:	LL10_TIscG
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	2499
Total relevant shots:	1445
Total relevant shots returned:	26
Precision at total relevant shots:	0.0105
Mean average precision:	0.0038



Interpolated recall	precision
0.0	0.0692
0.1	0.0056
0.2	0.0000
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

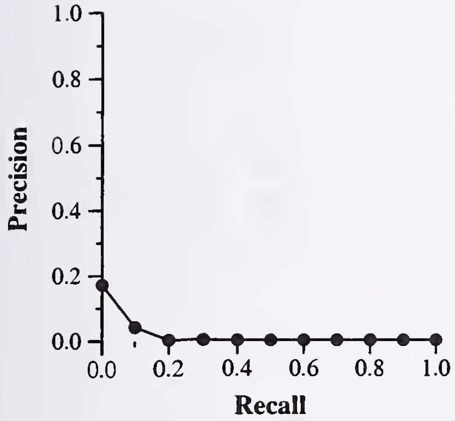
Precision at n shots	
5	0.0160
10	0.0120
15	0.0160
20	0.0160
30	0.0160
100	0.0104



Run score (dot) versus median (---) versus best (box) by topic

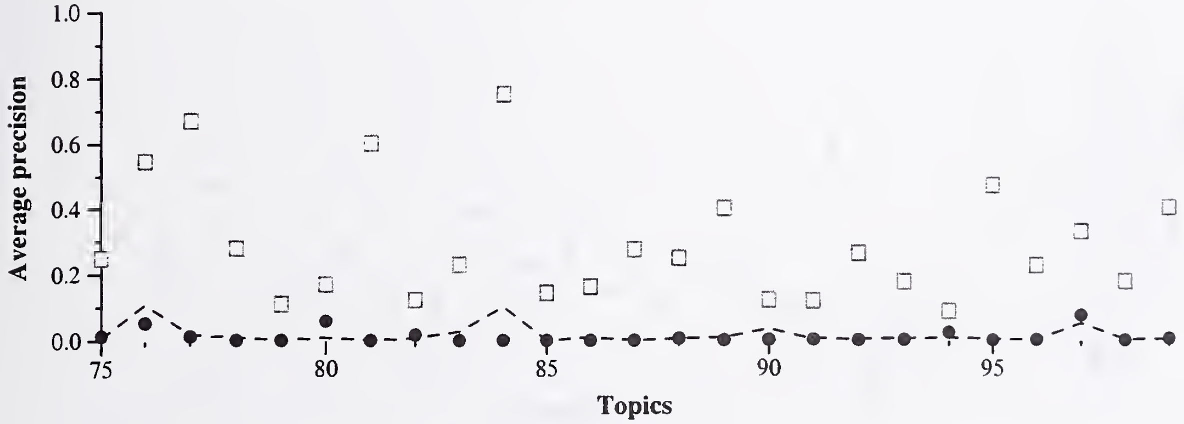
TREC 2002 Video Track: search results

Run ID:	MSRA.Q-Video.3
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	1245
Total relevant shots:	1445
Total relevant shots returned:	60
Precision at total relevant shots:	0.0249
Mean average precision:	0.0104



Interpolated recall	precision
0.0	0.1723
0.1	0.0404
0.2	0.0012
0.3	0.0012
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

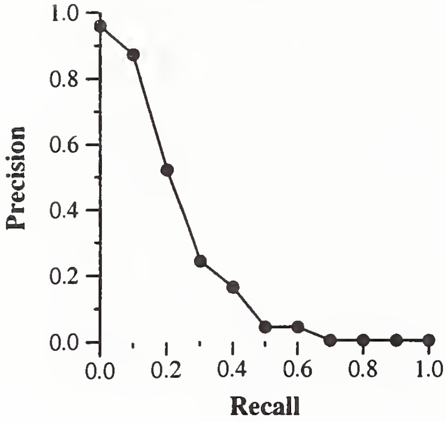
Precision at n shots	
5	0.0960
10	0.1000
15	0.0827
20	0.0780
30	0.0613
100	0.0240



Run score (dot) versus median (---) versus best (box) by topic

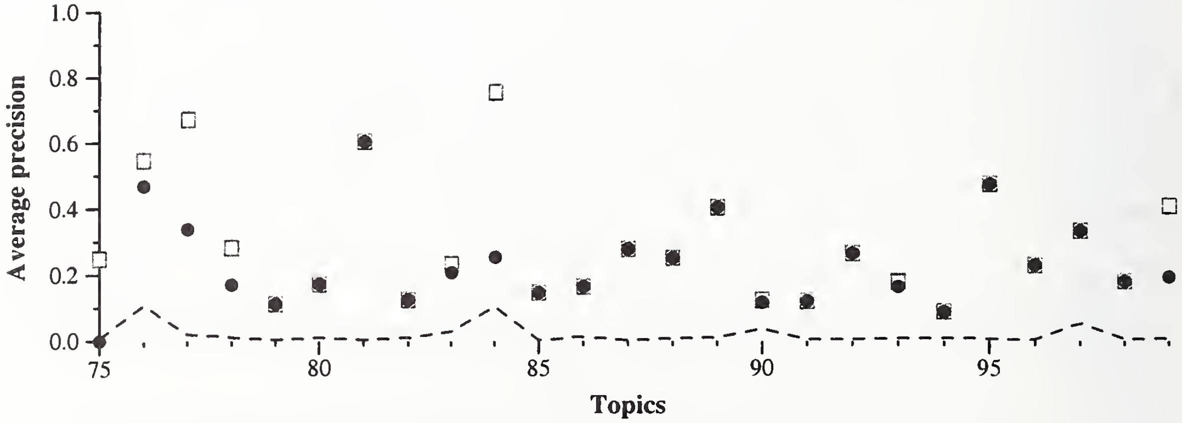
TREC 2002 Video Track: search results

Run ID:	ProusSci
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	307
Total relevant shots:	1445
Total relevant shots returned:	273
Precision at total relevant shots:	0.2421
Mean average precision:	0.2313



Interpolated recall	precision
0.0	0.9600
0.1	0.8726
0.2	0.5194
0.3	0.2386
0.4	0.1600
0.5	0.0400
0.6	0.0400
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

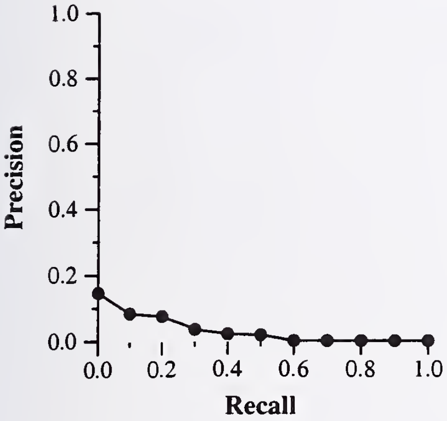
Precision at n shots	
5	0.7440
10	0.6360
15	0.5413
20	0.4780
30	0.3600
100	0.1092



Run score (dot) versus median (---) versus best (box) by topic

TREC 2002 Video Track: search results

Run ID:	UMDMNAqt
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	1862
Total relevant shots:	1445
Total relevant shots returned:	91
Precision at total relevant shots:	0.0646
Mean average precision:	0.0259

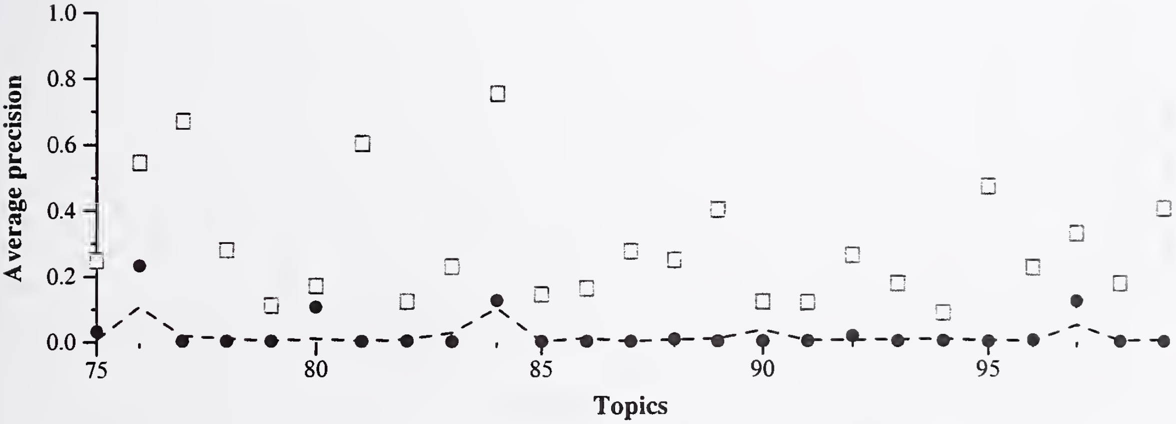


Interpolated recall precision

recall	precision
0.0	0.1472
0.1	0.0835
0.2	0.0755
0.3	0.0339
0.4	0.0205
0.5	0.0175
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

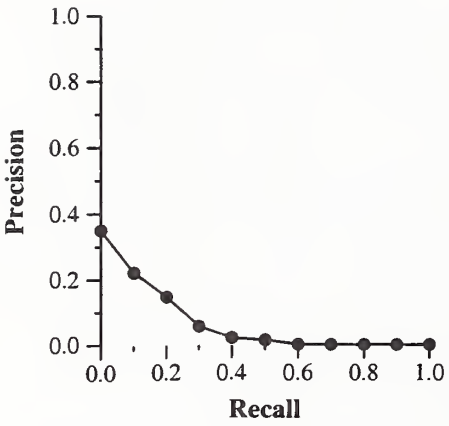
Precision at n shots

n	shots
5	0.0480
10	0.0480
15	0.0613
20	0.0680
30	0.0760
100	0.0364



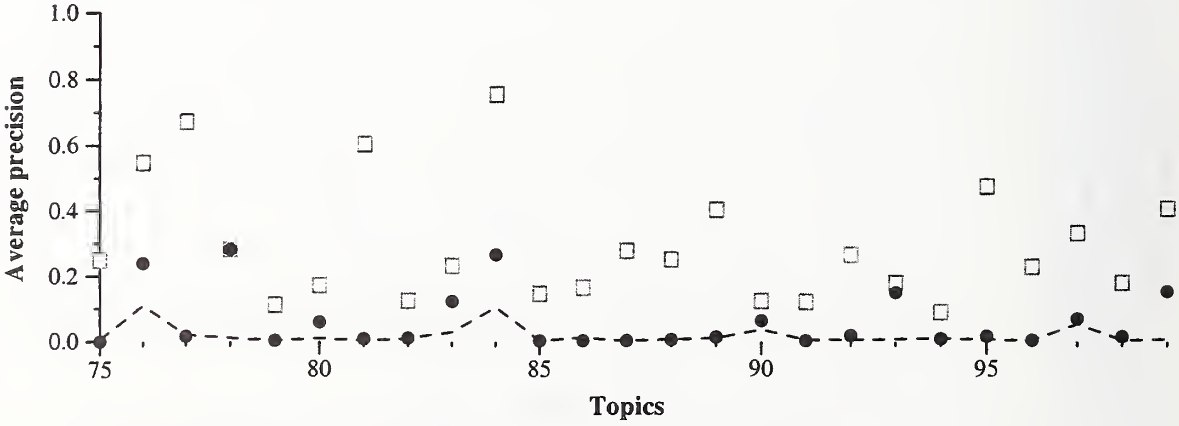
Run score (dot) versus median (---) versus best (box) by topic

```
TREC 2002 Video Track: search results

Run ID:                              UMDMqtrec
Processing type:                     manual
System training type:                B (no knowledge of test set)

Topics summarized:                   25

Total shots returned:                2364
Total relevant shots:                1445
Total relevant shots returned:       171

Precision at total relevant shots:   0.1103

Mean average precision:              0.0587
```



Interpolated recall	precision
0.0	0.3503
0.1	0.2208
0.2	0.1461
0.3	0.0546
0.4	0.0203
0.5	0.0135
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

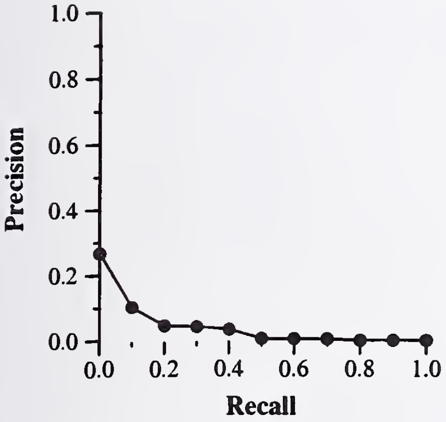
Precision at n shots	
5	0.1760
10	0.1520
15	0.1253
20	0.1140
30	0.1093
100	0.0684



Run score (dot) versus median (---) versus best (box) by topic

TREC 2002 Video Track: search results

Run ID:	UnivO_MT1
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	1330
Total relevant shots:	1445
Total relevant shots returned:	89
Precision at total relevant shots:	0.0521
Mean average precision:	0.0335

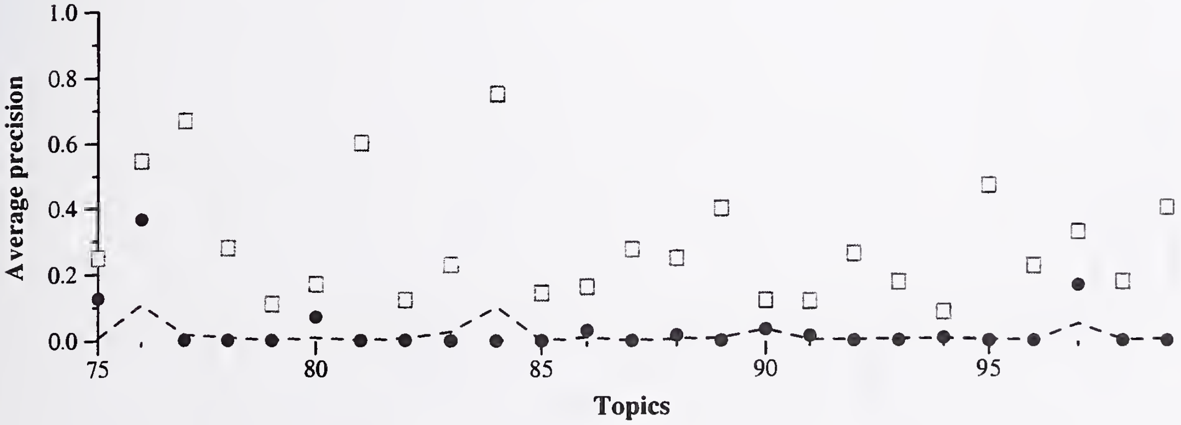


Interpolated recall precision

recall	precision
0.0	0.2674
0.1	0.1022
0.2	0.0450
0.3	0.0407
0.4	0.0336
0.5	0.0056
0.6	0.0048
0.7	0.0046
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots

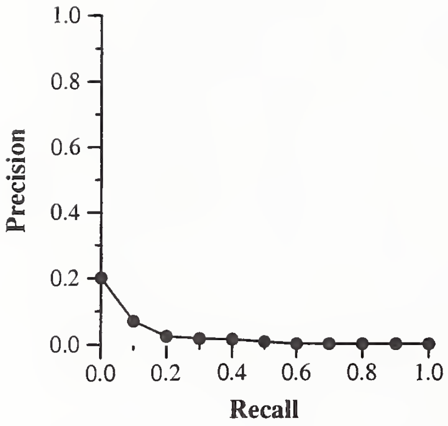
n	shots
5	0.1760
10	0.1520
15	0.1280
20	0.1060
30	0.0827
100	0.0356



Run score (dot) versus median (---) versus best (box) by topic

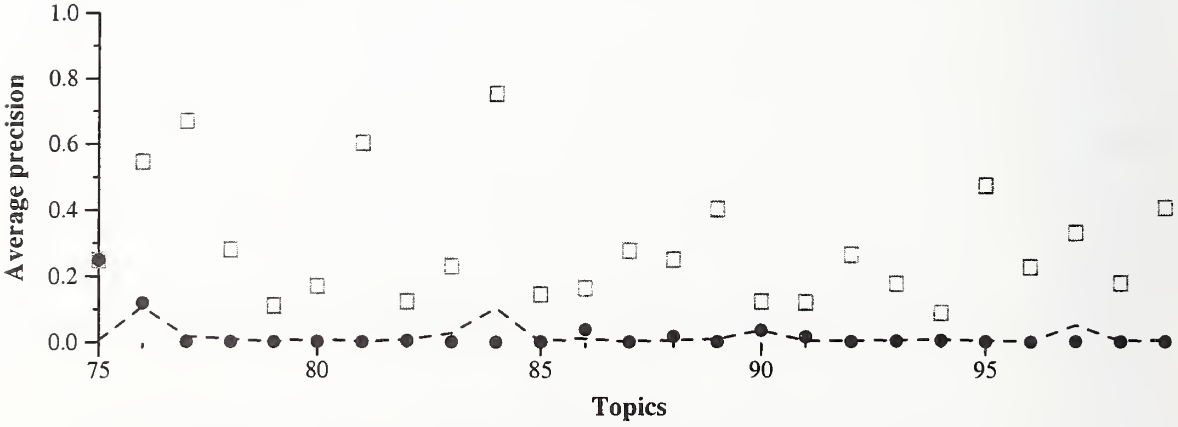
TREC 2002 Video Track: search results

Run ID:	UnivO_MT2
Processing type:	manual
System training type:	B (no knowledge of test set)
Topics summarized:	25
Total shots returned:	1066
Total relevant shots:	1445
Total relevant shots returned:	57
Precision at total relevant shots:	0.0368
Mean average precision:	0.0194



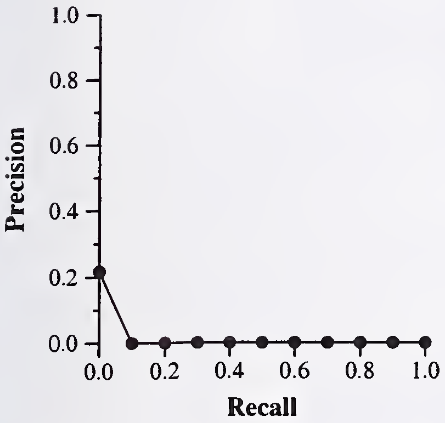
Interpolated recall	precision
0.0	0.2015
0.1	0.0702
0.2	0.0240
0.3	0.0154
0.4	0.0133
0.5	0.0064
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.1280
10	0.0880
15	0.0773
20	0.0660
30	0.0520
100	0.0228



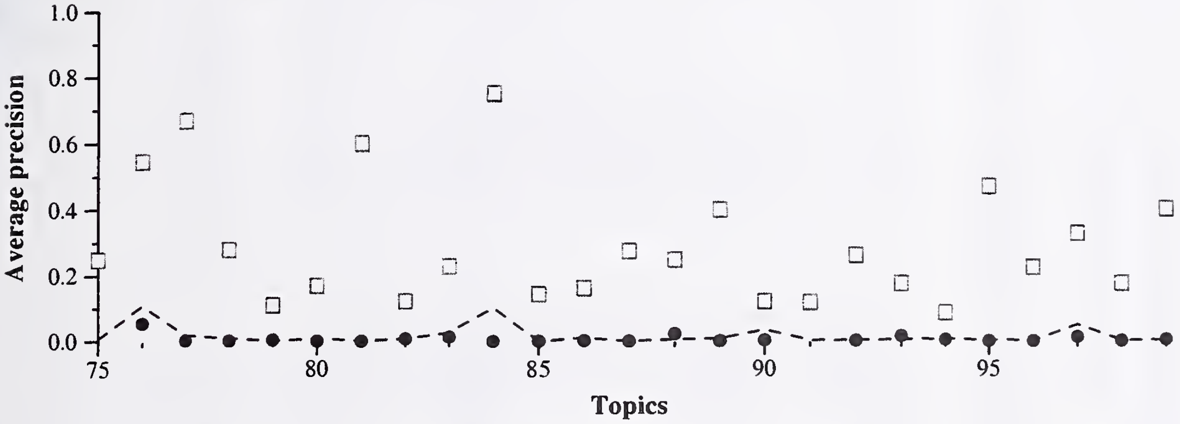
Run score (dot) versus median (---) versus best (box) by topic

```
TREC 2002 Video Track: search results

Run ID:                              UnivO_MT3
Processing type:                     manual
System training type:                B (no knowledge of test set)

Topics summarized:                   23

Total shots returned:                690
Total relevant shots:                1413
Total relevant shots returned:       43

Precision at total relevant shots:   0.0197

Mean average precision:              0.0061
```



Interpolated recall	precision
0.0	0.2172
0.1	0.0000
0.2	0.0000
0.3	0.0000
0.4	0.0000
0.5	0.0000
0.6	0.0000
0.7	0.0000
0.8	0.0000
0.9	0.0000
1.0	0.0000

Precision at n shots	
5	0.0696
10	0.0522
15	0.0522
20	0.0500
30	0.0406
100	0.0187



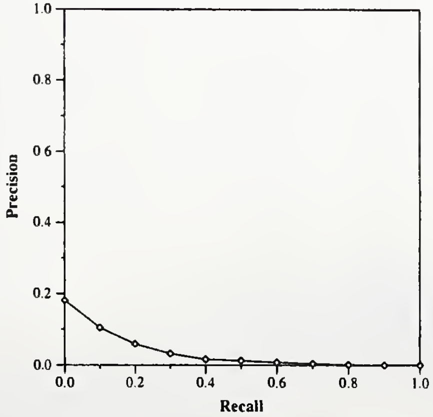
Run score (dot) versus median (---) versus best (box) by topic

Summary Statistics	
Run ID:	ajouai0206
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

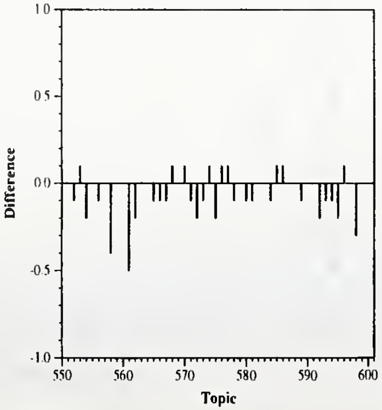
Total number of documents over all topics	
Retrieved:	48295
Relevant:	1574
Rel-ret:	470

Recall Level Averages	
Recall	Precision
0.00	0.1814
0.10	0.1043
0.20	0.0586
0.30	0.0318
0.40	0.0159
0.50	0.0124
0.60	0.0076
0.70	0.0042
0.80	0.0014
0.90	0.0002
1.00	0.0002
Mean average precision	
non-interpolated	0.0277

Document Level Averages	
	Precision
At 5 docs	0.0857
At 10 docs	0.0551
At 15 docs	0.0490
At 20 docs	0.0408
At 30 docs	0.0327
At 100 docs	0.0306
At 200 docs	0.0211
At 500 docs	0.0135
At 1000 docs	0.0096
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0461



Recall-Precision Curve



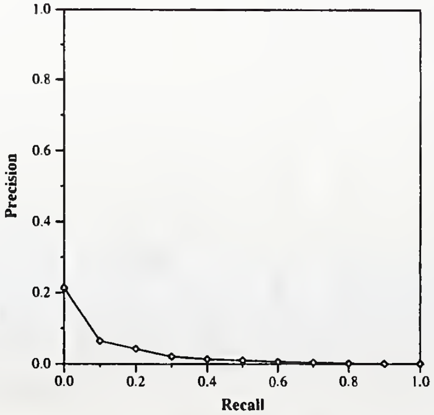
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	ajouai0207
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

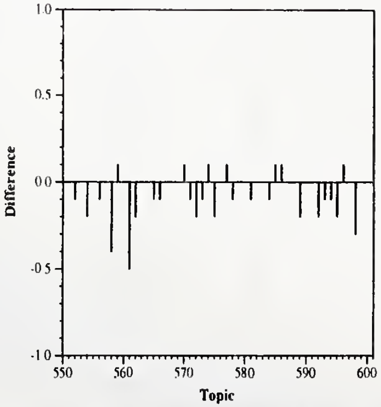
Total number of documents over all topics	
Retrieved:	48295
Relevant:	1574
Rel-ret:	456

Recall Level Averages	
Recall	Precision
0.00	0.2136
0.10	0.0644
0.20	0.0425
0.30	0.0209
0.40	0.0134
0.50	0.0101
0.60	0.0060
0.70	0.0038
0.80	0.0012
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0210

Document Level Averages	
	Precision
At 5 docs	0.0776
At 10 docs	0.0531
At 15 docs	0.0381
At 20 docs	0.0347
At 30 docs	0.0252
At 100 docs	0.0292
At 200 docs	0.0192
At 500 docs	0.0127
At 1000 docs	0.0093
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0362



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

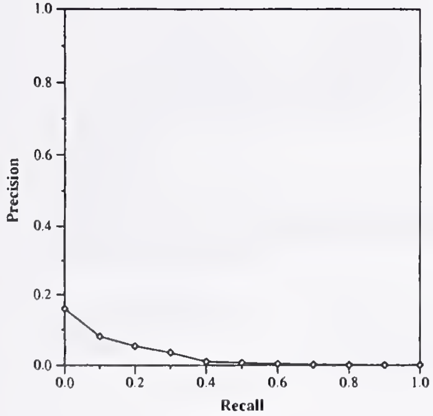
Web track, topic distillation task results — Ajou University

Summary Statistics	
Run ID:	ajouai0208
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

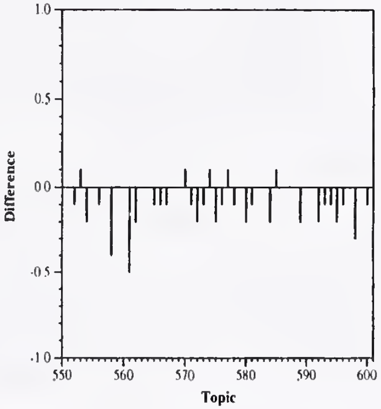
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	279

Recall Level Averages	
Recall	Precision
0.00	0.1596
0.10	0.0811
0.20	0.0534
0.30	0.0355
0.40	0.0106
0.50	0.0069
0.60	0.0038
0.70	0.0016
0.80	0.0002
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0227

Document Level Averages	
	Precision
At 5 docs	0.0531
At 10 docs	0.0347
At 15 docs	0.0272
At 20 docs	0.0265
At 30 docs	0.0272
At 100 docs	0.0229
At 200 docs	0.0153
At 500 docs	0.0085
At 1000 docs	0.0057
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0361



Recall-Precision Curve



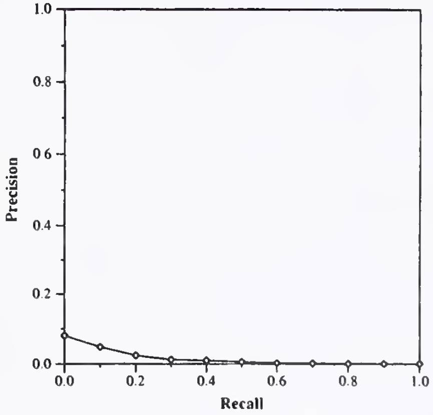
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	ajouai0209
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

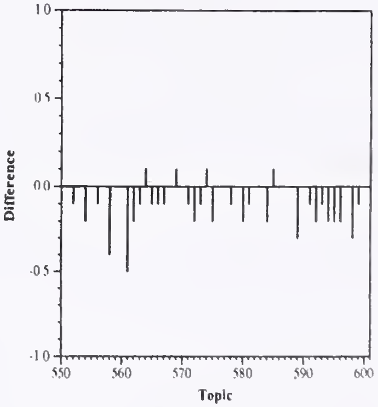
Total number of documents over all topics	
Retrieved:	48813
Relevant:	1574
Rel-ret:	218

Recall Level Averages	
Recall	Precision
0.00	0.0804
0.10	0.0479
0.20	0.0239
0.30	0.0125
0.40	0.0094
0.50	0.0060
0.60	0.0025
0.70	0.0015
0.80	0.0002
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0128

Document Level Averages	
	Precision
At 5 docs	0.0490
At 10 docs	0.0245
At 15 docs	0.0190
At 20 docs	0.0184
At 30 docs	0.0150
At 100 docs	0.0096
At 200 docs	0.0076
At 500 docs	0.0057
At 1000 docs	0.0044
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0180



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	ajouai0210
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

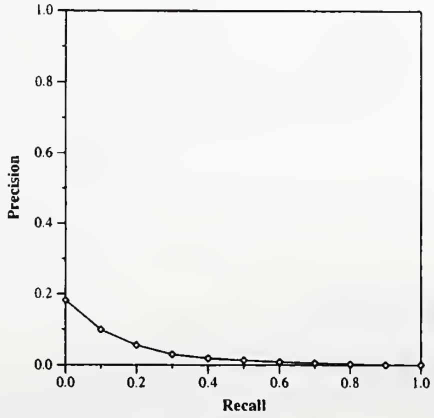
Total number of documents over all topics	
Retrieved:	48295
Relevant:	1574
Rel-ret:	484

Recall Level Averages	
Recall	Precision
0.00	0.1814
0.10	0.0986
0.20	0.0545
0.30	0.0293
0.40	0.0185
0.50	0.0130
0.60	0.0084
0.70	0.0049
0.80	0.0016
0.90	0.0001
1.00	0.0001

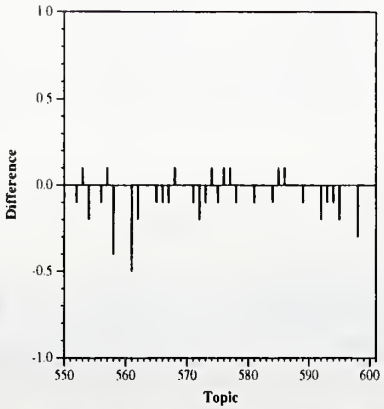
Mean average precision	
non-interpolated	0.0271

Document Level Averages	
	Precision
At 5 docs	0.0857
At 10 docs	0.0571
At 15 docs	0.0490
At 20 docs	0.0469
At 30 docs	0.0388
At 100 docs	0.0300
At 200 docs	0.0203
At 500 docs	0.0141
At 1000 docs	0.0099

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0403



Recall-Precision Curve



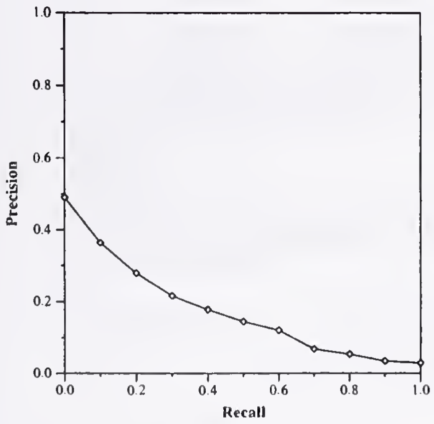
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	icttd1
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

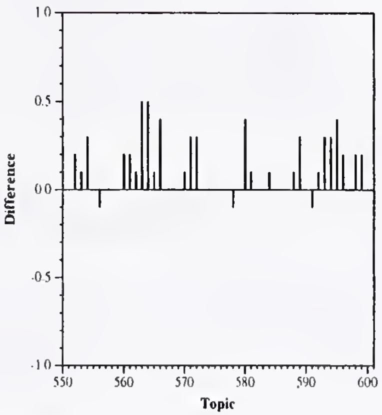
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	1038

Recall Level Averages	
Recall	Precision
0.00	0.4904
0.10	0.3638
0.20	0.2792
0.30	0.2160
0.40	0.1782
0.50	0.1453
0.60	0.1211
0.70	0.0684
0.80	0.0537
0.90	0.0347
1.00	0.0293
Mean average precision	
non-interpolated	0.1620

Document Level Averages	
	Precision
At 5 docs	0.2653
At 10 docs	0.2306
At 15 docs	0.1959
At 20 docs	0.1806
At 30 docs	0.1497
At 100 docs	0.0939
At 200 docs	0.0656
At 500 docs	0.0356
At 1000 docs	0.0212
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1919



Recall-Precision Curve



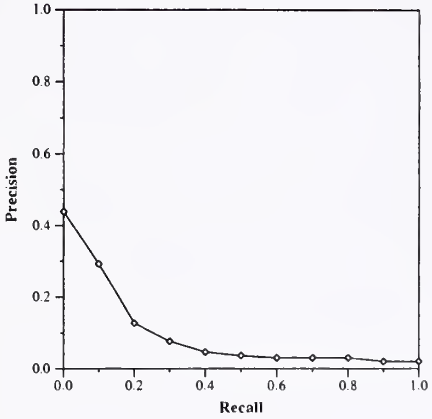
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	icttd2
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

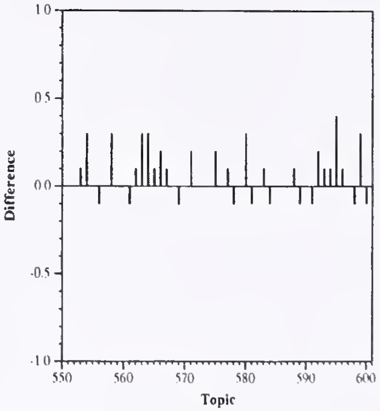
Total number of documents over all topics	
Retrieved:	9065
Relevant:	1574
Rel-ret:	288

Recall Level Averages	
Recall	Precision
0.00	0.4388
0.10	0.2923
0.20	0.1279
0.30	0.0776
0.40	0.0469
0.50	0.0367
0.60	0.0306
0.70	0.0306
0.80	0.0306
0.90	0.0204
1.00	0.0204
Mean average precision	
non-interpolated	0.0868

Document Level Averages	
	Precision
At 5 docs	0.2082
At 10 docs	0.1755
At 15 docs	0.1497
At 20 docs	0.1367
At 30 docs	0.1163
At 100 docs	0.0531
At 200 docs	0.0291
At 500 docs	0.0118
At 1000 docs	0.0059
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1237



Recall-Precision Curve



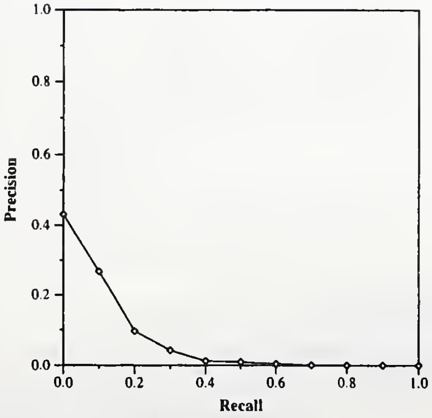
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	icttd3
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

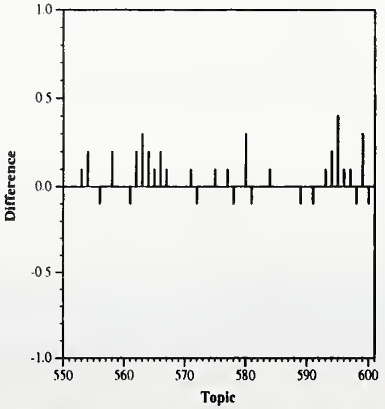
Total number of documents over all topics	
Retrieved:	9082
Relevant:	1574
Rel-ret:	288

Recall Level Averages	
Recall	Precision
0.00	0.4310
0.10	0.2676
0.20	0.0965
0.30	0.0421
0.40	0.0124
0.50	0.0094
0.60	0.0038
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0584

Document Level Averages	
	Precision
At 5 docs	0.2163
At 10 docs	0.1673
At 15 docs	0.1442
At 20 docs	0.1327
At 30 docs	0.1156
At 100 docs	0.0531
At 200 docs	0.0291
At 500 docs	0.0118
At 1000 docs	0.0059
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1034



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	pltr02wt1
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

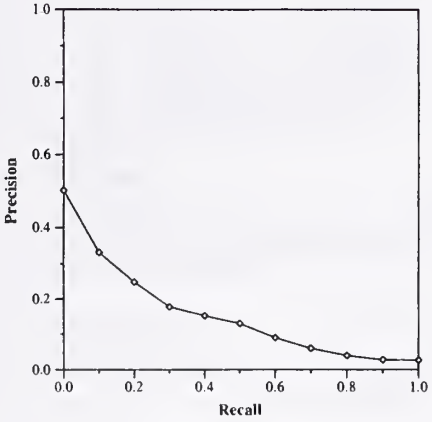
Total number of documents over all topics	
Retrieved:	48210
Relevant:	1574
Rel-ret:	1002

Recall Level Averages	
Recall	Precision
0.00	0.5017
0.10	0.3320
0.20	0.2483
0.30	0.1777
0.40	0.1523
0.50	0.1299
0.60	0.0898
0.70	0.0599
0.80	0.0397
0.90	0.0278
1.00	0.0266

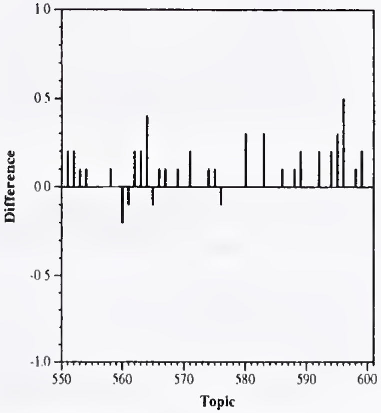
Mean average precision	
non-interpolated	0.1438

Document Level Averages	
	Precision
At 5 docs	0.2367
At 10 docs	0.2000
At 15 docs	0.1728
At 20 docs	0.1571
At 30 docs	0.1374
At 100 docs	0.0918
At 200 docs	0.0663
At 500 docs	0.0351
At 1000 docs	0.0204

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1721



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	pltr02wt2
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

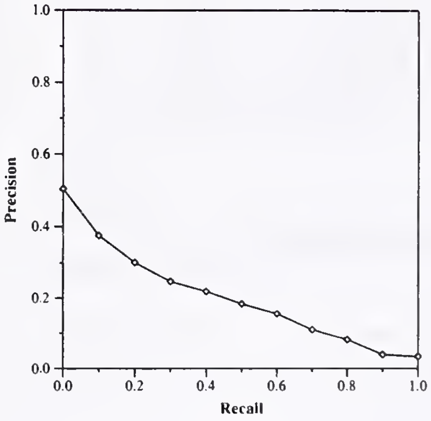
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	1101

Recall Level Averages	
Recall	Precision
0.00	0.5036
0.10	0.3753
0.20	0.3008
0.30	0.2475
0.40	0.2191
0.50	0.1837
0.60	0.1551
0.70	0.1098
0.80	0.0821
0.90	0.0392
1.00	0.0339

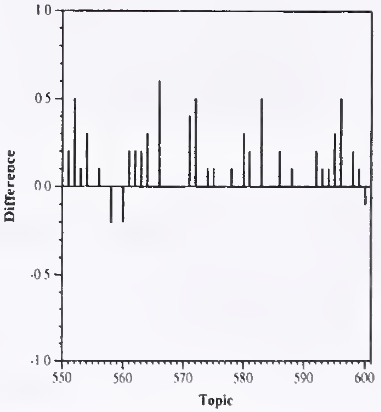
Mean average precision	
non-interpolated	0.1896

Document Level Averages	
	Precision
At 5 docs	0.2776
At 10 docs	0.2408
At 15 docs	0.2068
At 20 docs	0.1929
At 30 docs	0.1673
At 100 docs	0.1059
At 200 docs	0.0742
At 500 docs	0.0391
At 1000 docs	0.0225

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.2153



Recall-Precision Curve



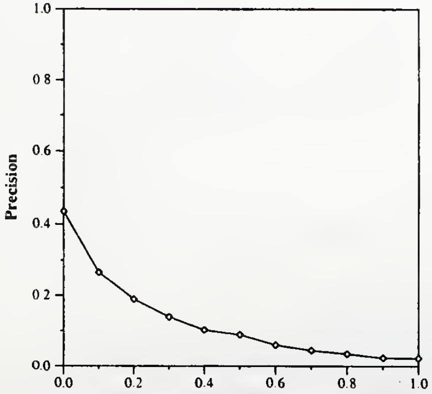
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	pltr02wt3
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

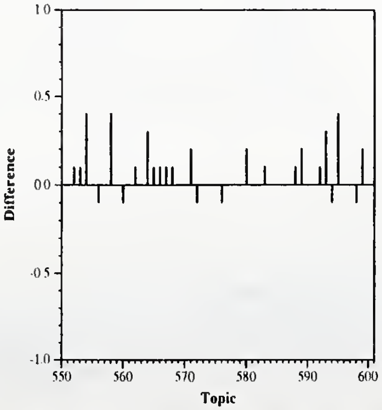
Total number of documents over all topics	
Retrieved:	48389
Relevant:	1574
Rel-ret:	955

Recall Level Averages	
Recall	Precision
0.00	0.4346
0.10	0.2647
0.20	0.1892
0.30	0.1397
0.40	0.1026
0.50	0.0892
0.60	0.0601
0.70	0.0447
0.80	0.0349
0.90	0.0237
1.00	0.0226
Mean average precision	
non-interpolated	0.1088

Document Level Averages	
	Precision
At 5 docs	0.2245
At 10 docs	0.1755
At 15 docs	0.1442
At 20 docs	0.1214
At 30 docs	0.1143
At 100 docs	0.0712
At 200 docs	0.0536
At 500 docs	0.0313
At 1000 docs	0.0195
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1450



Recall-Precision Curve



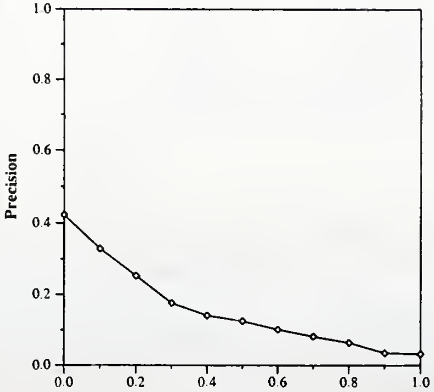
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	pltr02wt4
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

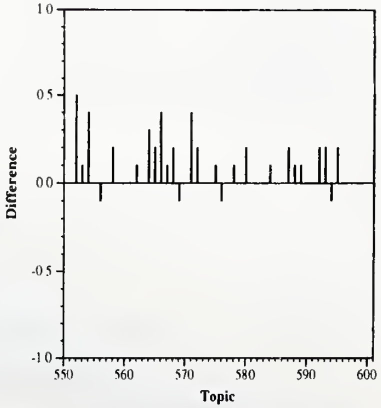
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	1039

Recall Level Averages	
Recall	Precision
0.00	0.4222
0.10	0.3285
0.20	0.2528
0.30	0.1762
0.40	0.1411
0.50	0.1252
0.60	0.1015
0.70	0.0813
0.80	0.0641
0.90	0.0348
1.00	0.0320
Mean average precision	
non-interpolated	0.1426

Document Level Averages	
	Precision
At 5 docs	0.2327
At 10 docs	0.2000
At 15 docs	0.1701
At 20 docs	0.1541
At 30 docs	0.1354
At 100 docs	0.0796
At 200 docs	0.0604
At 500 docs	0.0344
At 1000 docs	0.0212
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1652



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Web track, topic distillation task results — City University-London

Summary Statistics	
Run ID:	pltr02wt5
Run Description	Exploratory; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

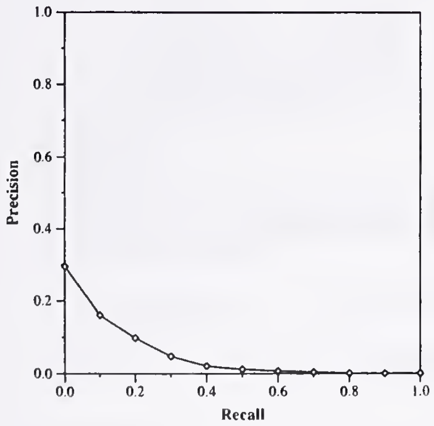
Total number of documents over all topics	
Retrieved:	48475
Relevant:	1574
Rel-ret:	502

Recall Level Averages	
Recall	Precision
0.00	0.2956
0.10	0.1608
0.20	0.0984
0.30	0.0479
0.40	0.0209
0.50	0.0122
0.60	0.0070
0.70	0.0040
0.80	0.0005
0.90	0.0004
1.00	0.0004

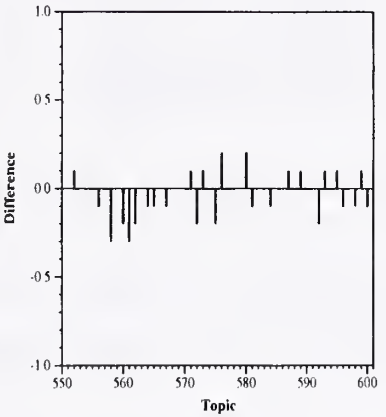
Mean average precision	
non-interpolated	0.0445

Document Level Averages	
	Precision
At 5 docs	0.1143
At 10 docs	0.0878
At 15 docs	0.0816
At 20 docs	0.0724
At 30 docs	0.0673
At 100 docs	0.0400
At 200 docs	0.0270
At 500 docs	0.0160
At 1000 docs	0.0102

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0740



Recall-Precision Curve



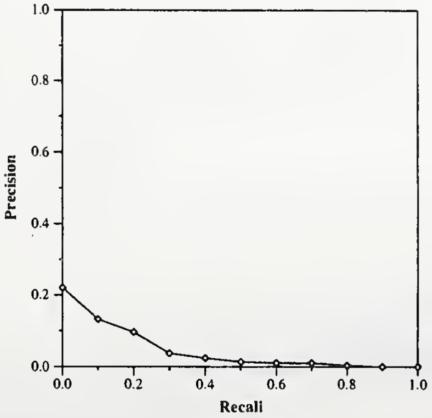
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	csiro02td1
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

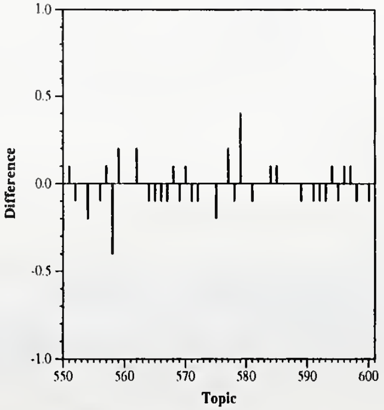
Total number of documents over all topics	
Retrieved:	46682
Relevant:	1574
Rel-ret:	313

Recall Level Averages	
Recall	Precision
0.00	0.2201
0.10	0.1318
0.20	0.0963
0.30	0.0373
0.40	0.0244
0.50	0.0138
0.60	0.0104
0.70	0.0104
0.80	0.0034
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0405

Document Level Averages	
	Precision
At 5 docs	0.0980
At 10 docs	0.1000
At 15 docs	0.0816
At 20 docs	0.0704
At 30 docs	0.0646
At 100 docs	0.0357
At 200 docs	0.0222
At 500 docs	0.0114
At 1000 docs	0.0064
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0824



Recall-Precision Curve



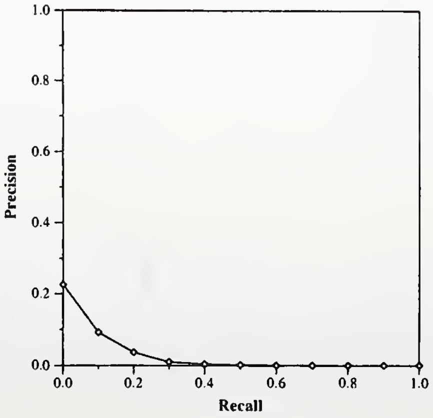
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	csiro02td2
Run Description	Realistic; NODOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

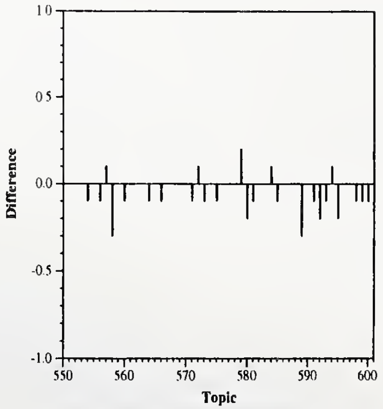
Total number of documents over all topics	
Retrieved:	48119
Relevant:	1574
Rel-ret:	278

Recall Level Averages	
Recall	Precision
0.00	0.2261
0.10	0.0923
0.20	0.0371
0.30	0.0103
0.40	0.0041
0.50	0.0022
0.60	0.0006
0.70	0.0001
0.80	0.0000
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0239

Document Level Averages	
	Precision
At 5 docs	0.0898
At 10 docs	0.0714
At 15 docs	0.0612
At 20 docs	0.0551
At 30 docs	0.0476
At 100 docs	0.0253
At 200 docs	0.0179
At 500 docs	0.0095
At 1000 docs	0.0057
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0460



Recall-Precision Curve



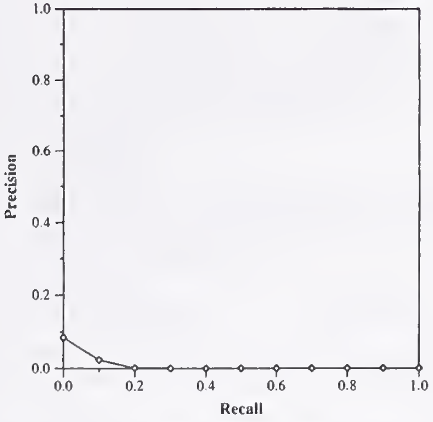
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	csiro02td3
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

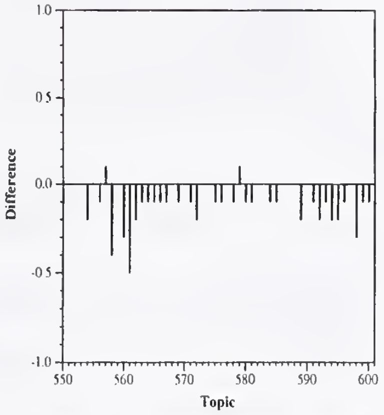
Total number of documents over all topics	
Retrieved:	13482
Relevant:	1574
Rel-ret:	71

Recall Level Averages	
Recall	Precision
0.00	0.0849
0.10	0.0235
0.20	0.0010
0.30	0.0000
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0050

Document Level Averages	
	Precision
At 5 docs	0.0204
At 10 docs	0.0184
At 15 docs	0.0136
At 20 docs	0.0133
At 30 docs	0.0116
At 100 docs	0.0088
At 200 docs	0.0058
At 500 docs	0.0029
At 1000 docs	0.0014
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0112



Recall-Precision Curve



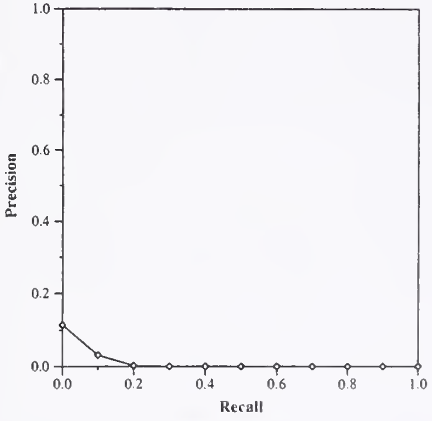
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	csiro02td4
Run Description	Realistic; NODOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

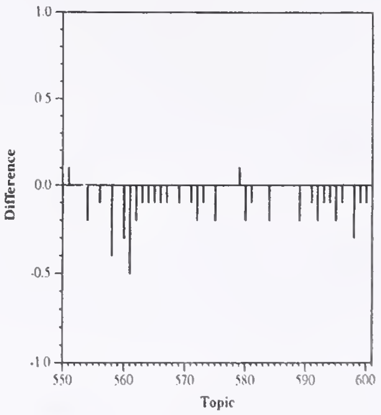
Total number of documents over all topics	
Retrieved:	20723
Relevant:	1574
Rel-ret:	84

Recall Level Averages	
Recall	Precision
0.00	0.1144
0.10	0.0314
0.20	0.0021
0.30	0.0006
0.40	0.0003
0.50	0.0003
0.60	0.0003
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0073

Document Level Averages	
	Precision
At 5 docs	0.0163
At 10 docs	0.0184
At 15 docs	0.0218
At 20 docs	0.0184
At 30 docs	0.0190
At 100 docs	0.0120
At 200 docs	0.0078
At 500 docs	0.0034
At 1000 docs	0.0017
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0230



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Web track, topic distillation task results — CSIRO

Summary Statistics	
Run ID:	csiro02td5
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

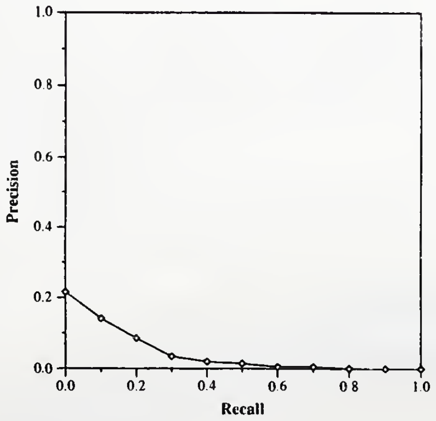
Total number of documents over all topics	
Retrieved:	47076
Relevant:	1574
Rel-ret:	315

Recall Level Averages	
Recall	Precision
0.00	0.2160
0.10	0.1405
0.20	0.0852
0.30	0.0342
0.40	0.0195
0.50	0.0142
0.60	0.0048
0.70	0.0048
0.80	0.0000
0.90	0.0000
1.00	0.0000

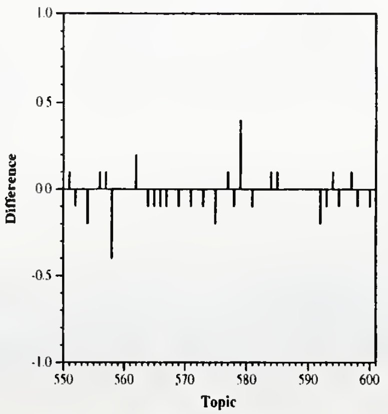
Mean average precision	
non-interpolated	0.0383

Document Level Averages	
	Precision
At 5 docs	0.0816
At 10 docs	0.0939
At 15 docs	0.0884
At 20 docs	0.0776
At 30 docs	0.0701
At 100 docs	0.0388
At 200 docs	0.0229
At 500 docs	0.0117
At 1000 docs	0.0064

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0805



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

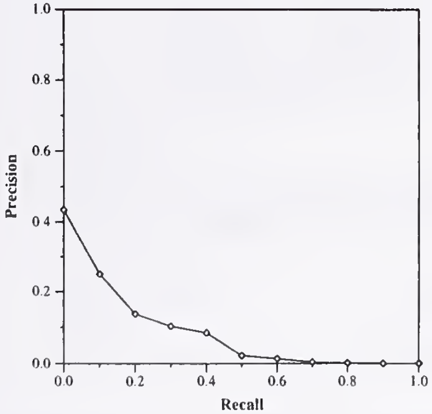
Web track, topic distillation task results — Fudan University

Summary Statistics	
Run ID:	fduwt11b0
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

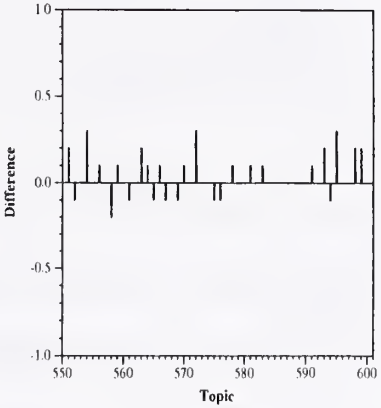
Total number of documents over all topics	
Retrieved:	40436
Relevant:	1574
Rel-ret:	578

Recall Level Averages	
Recall	Precision
0.00	0.4343
0.10	0.2493
0.20	0.1382
0.30	0.1036
0.40	0.0855
0.50	0.0225
0.60	0.0135
0.70	0.0039
0.80	0.0014
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0749

Document Level Averages	
	Precision
At 5 docs	0.1878
At 10 docs	0.1510
At 15 docs	0.1401
At 20 docs	0.1296
At 30 docs	0.1102
At 100 docs	0.0582
At 200 docs	0.0373
At 500 docs	0.0204
At 1000 docs	0.0118
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1309



Recall-Precision Curve



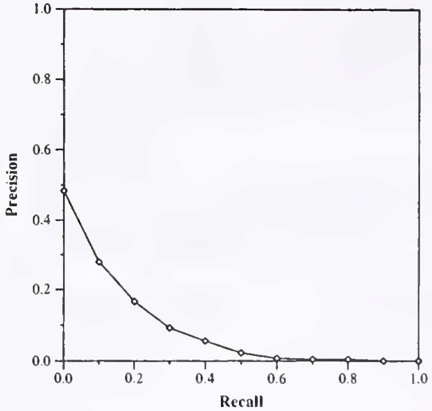
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	fduwt11o1
Run Description	Realistic; DOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

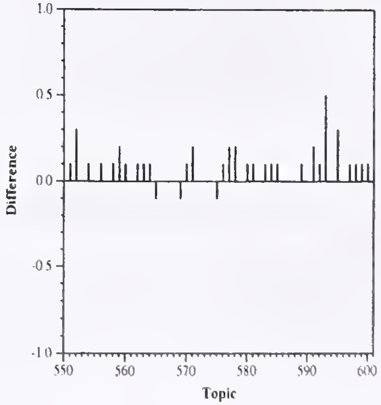
Total number of documents over all topics	
Retrieved:	1470
Relevant:	1574
Rel-ret:	175

Recall Level Averages	
Recall	Precision
0.00	0.4837
0.10	0.2787
0.20	0.1663
0.30	0.0926
0.40	0.0561
0.50	0.0230
0.60	0.0077
0.70	0.0048
0.80	0.0048
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0818

Document Level Averages	
	Precision
At 5 docs	0.2571
At 10 docs	0.1939
At 15 docs	0.1633
At 20 docs	0.1449
At 30 docs	0.1190
At 100 docs	0.0357
At 200 docs	0.0179
At 500 docs	0.0071
At 1000 docs	0.0036
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1338



Recall-Precision Curve



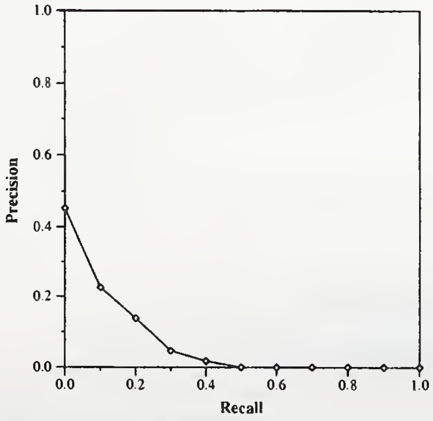
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	fduwt11o2
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

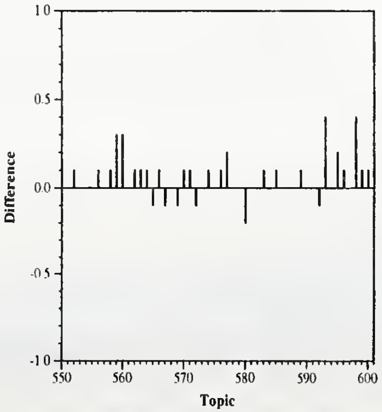
Total number of documents over all topics	
Retrieved:	1470
Relevant:	1574
Rel-ret:	165

Recall Level Averages	
Recall	Precision
0.00	0.4527
0.10	0.2243
0.20	0.1382
0.30	0.0468
0.40	0.0178
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0600

Document Level Averages	
	Precision
At 5 docs	0.2122
At 10 docs	0.1714
At 15 docs	0.1510
At 20 docs	0.1337
At 30 docs	0.1122
At 100 docs	0.0337
At 200 docs	0.0168
At 500 docs	0.0067
At 1000 docs	0.0034
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1110



Recall-Precision Curve



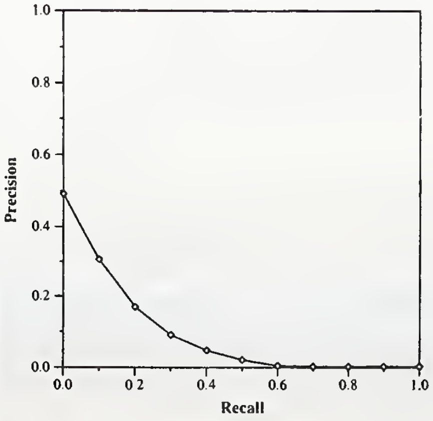
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	fduwt11t1
Run Description	Realistic; DOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

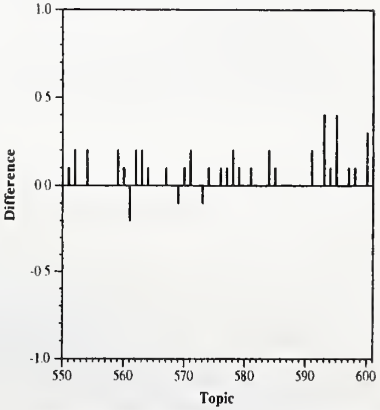
Total number of documents over all topics	
Retrieved:	9517
Relevant:	1574
Rel-ret:	285

Recall Level Averages	
Recall	Precision
0.00	0.4919
0.10	0.3061
0.20	0.1687
0.30	0.0905
0.40	0.0472
0.50	0.0212
0.60	0.0051
0.70	0.0027
0.80	0.0027
0.90	0.0027
1.00	0.0027
Mean average precision	
non-interpolated	0.0827

Document Level Averages	
	Precision
At 5 docs	0.2612
At 10 docs	0.1939
At 15 docs	0.1646
At 20 docs	0.1418
At 30 docs	0.1204
At 100 docs	0.0535
At 200 docs	0.0289
At 500 docs	0.0116
At 1000 docs	0.0058
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1420



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	fduwt11t2
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

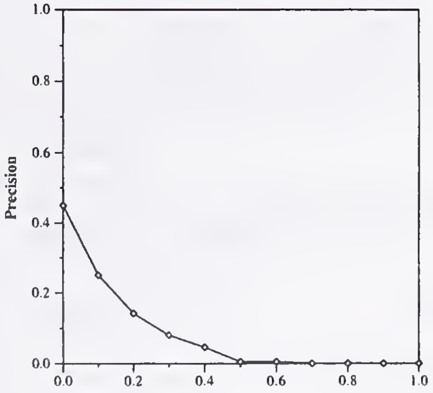
Total number of documents over all topics	
Retrieved:	1470
Relevant:	1574
Rel-ret:	169

Recall Level Averages	
Recall	Precision
0.00	0.4493
0.10	0.2508
0.20	0.1422
0.30	0.0810
0.40	0.0465
0.50	0.0051
0.60	0.0051
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000

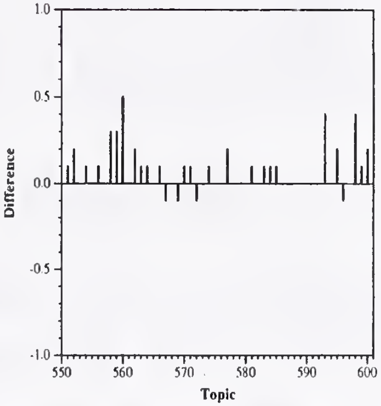
Mean average precision	
non-interpolated	0.0705

Document Level Averages	
	Precision
At 5 docs	0.2327
At 10 docs	0.1939
At 15 docs	0.1605
At 20 docs	0.1439
At 30 docs	0.1150
At 100 docs	0.0345
At 200 docs	0.0172
At 500 docs	0.0069
At 1000 docs	0.0034

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1295



Recall-Precision Curve



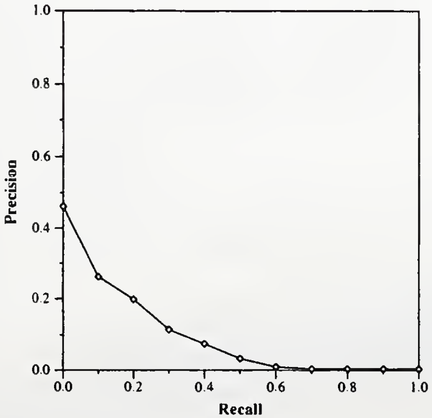
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	ibmHaifaAP
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

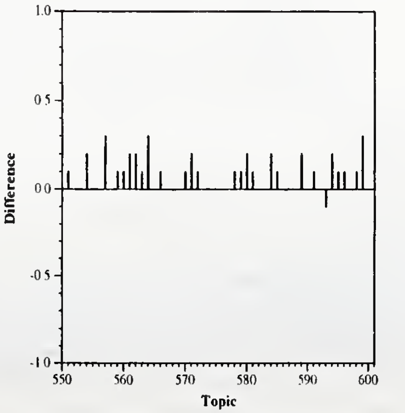
Total number of documents over all topics	
Retrieved:	4900
Relevant:	1574
Rel-ret:	363

Recall Level Averages	
Recall	Precision
0.00	0.4616
0.10	0.2615
0.20	0.1983
0.30	0.1151
0.40	0.0751
0.50	0.0335
0.60	0.0094
0.70	0.0034
0.80	0.0034
0.90	0.0034
1.00	0.0034
Mean average precision	
non-interpolated	0.0889

Document Level Averages	
	Precision
At 5 docs	0.2449
At 10 docs	0.1939
At 15 docs	0.1633
At 20 docs	0.1459
At 30 docs	0.1272
At 100 docs	0.0741
At 200 docs	0.0370
At 500 docs	0.0148
At 1000 docs	0.0074
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1416



Recall-Precision Curve



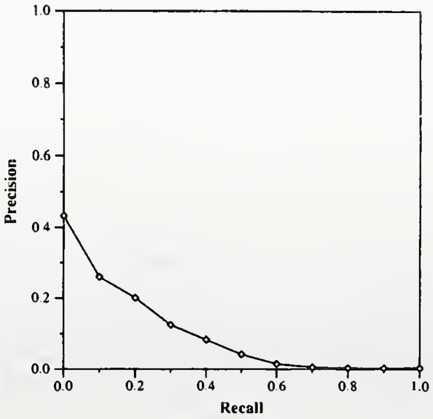
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	ibmHaifaBase
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

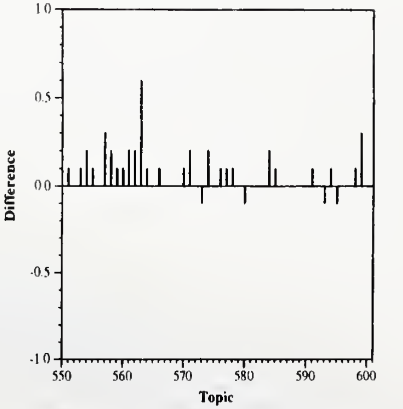
Total number of documents over all topics	
Retrieved:	4900
Relevant:	1574
Rel-ret:	383

Recall Level Averages	
Recall	Precision
0.00	0.4326
0.10	0.2594
0.20	0.2005
0.30	0.1254
0.40	0.0837
0.50	0.0423
0.60	0.0148
0.70	0.0056
0.80	0.0034
0.90	0.0034
1.00	0.0034
Mean average precision	
non-interpolated	0.0908

Document Level Averages	
	Precision
At 5 docs	0.2245
At 10 docs	0.1898
At 15 docs	0.1714
At 20 docs	0.1520
At 30 docs	0.1333
At 100 docs	0.0782
At 200 docs	0.0391
At 500 docs	0.0156
At 1000 docs	0.0078
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1444



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

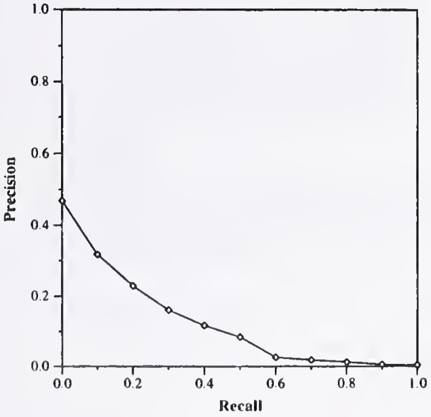
Web track, topic distillation task results — IBM Research Lab in Haifa

Summary Statistics	
Run ID:	ibmHaifaPR
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

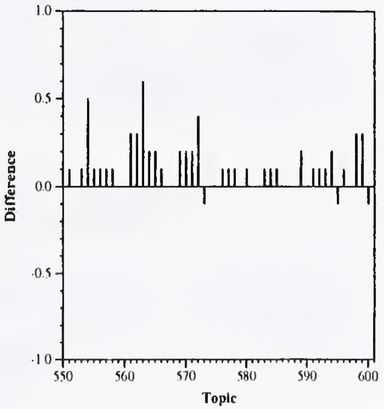
Total number of documents over all topics	
Retrieved:	9800
Relevant:	1574
Rel-ret:	623

Recall Level Averages	
Recall	Precision
0.00	0.4683
0.10	0.3175
0.20	0.2293
0.30	0.1607
0.40	0.1175
0.50	0.0846
0.60	0.0261
0.70	0.0184
0.80	0.0124
0.90	0.0056
1.00	0.0040
Mean average precision	
non-interpolated	0.1127

Document Level Averages	
	Precision
At 5 docs	0.2653
At 10 docs	0.2286
At 15 docs	0.1973
At 20 docs	0.1735
At 30 docs	0.1327
At 100 docs	0.0727
At 200 docs	0.0636
At 500 docs	0.0254
At 1000 docs	0.0127
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1653



Recall-Precision Curve



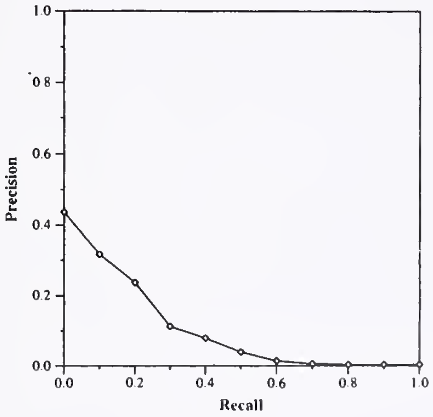
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	ibmHaifaT10
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

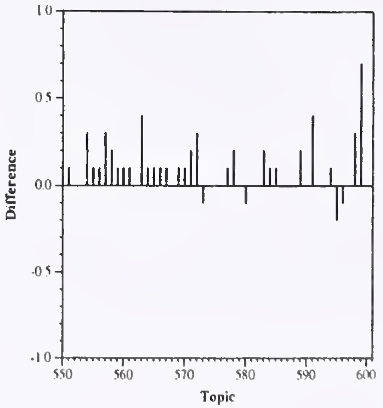
Total number of documents over all topics	
Retrieved:	4900
Relevant:	1574
Rel-ret:	390

Recall Level Averages	
Recall	Precision
0.00	0.4366
0.10	0.3168
0.20	0.2371
0.30	0.1131
0.40	0.0796
0.50	0.0396
0.60	0.0141
0.70	0.0056
0.80	0.0034
0.90	0.0034
1.00	0.0034
Mean average precision	
non-interpolated	0.0979

Document Level Averages	
	Precision
At 5 docs	0.2122
At 10 docs	0.2122
At 15 docs	0.1864
At 20 docs	0.1602
At 30 docs	0.1388
At 100 docs	0.0796
At 200 docs	0.0398
At 500 docs	0.0159
At 1000 docs	0.0080
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1565



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Web track, topic distillation task results — IBM Research Lab in Haifa

Summary Statistics	
Run ID:	ibmHaifaT10D
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

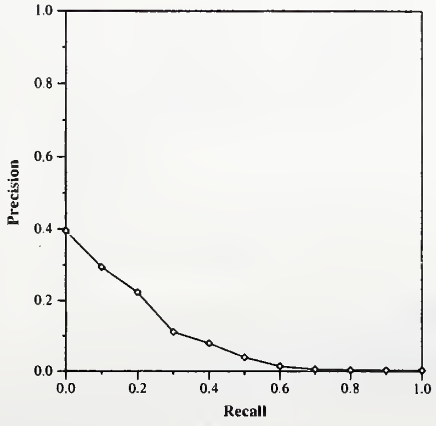
Total number of documents over all topics	
Retrieved:	4900
Relevant:	1574
Rel-ret:	387

Recall Level Averages	
Recall	Precision
0.00	0.3951
0.10	0.2945
0.20	0.2237
0.30	0.1114
0.40	0.0796
0.50	0.0396
0.60	0.0141
0.70	0.0056
0.80	0.0034
0.90	0.0034
1.00	0.0034

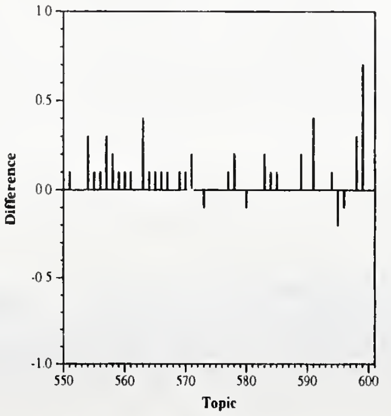
Mean average precision	
non-interpolated	0.0912

Document Level Averages	
	Precision
At 5 docs	0.2000
At 10 docs	0.2061
At 15 docs	0.1823
At 20 docs	0.1571
At 30 docs	0.1367
At 100 docs	0.0790
At 200 docs	0.0395
At 500 docs	0.0158
At 1000 docs	0.0079

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1537



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

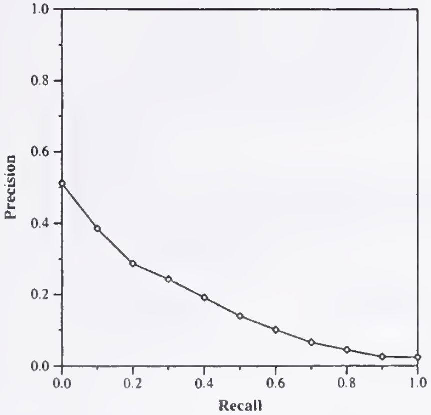
Web track, topic distillation task results — Institut de Recherche en Informatique de Toulouse (IRIT/SIG)

Summary Statistics	
Run ID:	Mercah
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

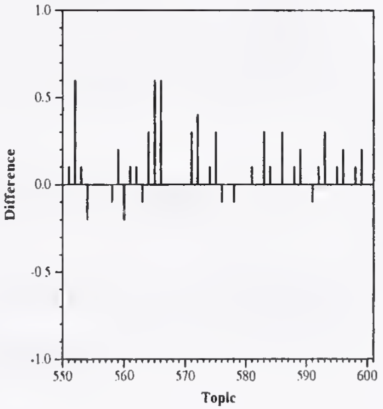
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	994

Recall Level Averages	
Recall	Precision
0.00	0.5115
0.10	0.3859
0.20	0.2879
0.30	0.2438
0.40	0.1921
0.50	0.1399
0.60	0.1012
0.70	0.0654
0.80	0.0451
0.90	0.0258
1.00	0.0241
Mean average precision	
non-interpolated	0.1645

Document Level Averages	
	Precision
At 5 docs	0.2449
At 10 docs	0.2163
At 15 docs	0.2041
At 20 docs	0.1765
At 30 docs	0.1463
At 100 docs	0.0898
At 200 docs	0.0661
At 500 docs	0.0356
At 1000 docs	0.0203
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1984



Recall-Precision Curve



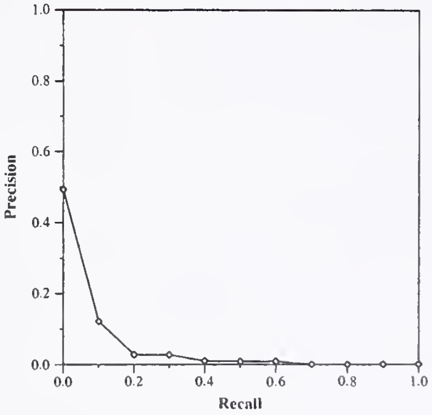
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	Mercure
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

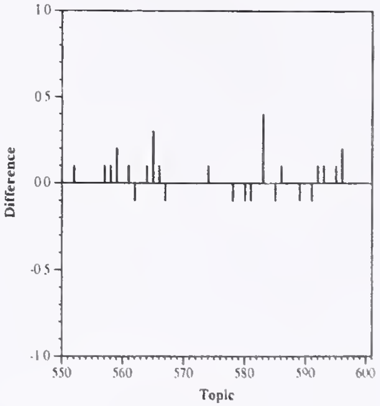
Total number of documents over all topics	
Retrieved:	415
Relevant:	1574
Rel-ret:	71

Recall Level Averages	
Recall	Precision
0.00	0.4929
0.10	0.1219
0.20	0.0277
0.30	0.0277
0.40	0.0102
0.50	0.0087
0.60	0.0087
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0414

Document Level Averages	
	Precision
At 5 docs	0.2041
At 10 docs	0.1429
At 15 docs	0.0966
At 20 docs	0.0724
At 30 docs	0.0483
At 100 docs	0.0145
At 200 docs	0.0072
At 500 docs	0.0029
At 1000 docs	0.0014
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0575



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	MercureLynx
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

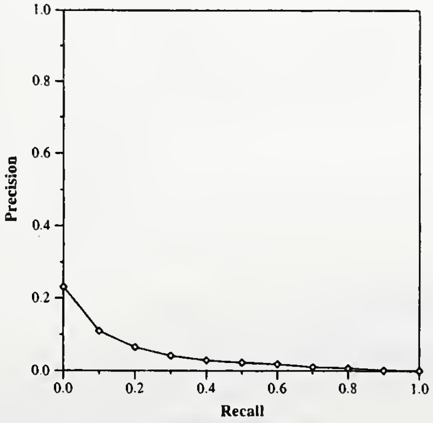
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	994

Recall Level Averages	
Recall	Precision
0.00	0.2308
0.10	0.1099
0.20	0.0653
0.30	0.0418
0.40	0.0289
0.50	0.0227
0.60	0.0182
0.70	0.0106
0.80	0.0074
0.90	0.0014
1.00	0.0004

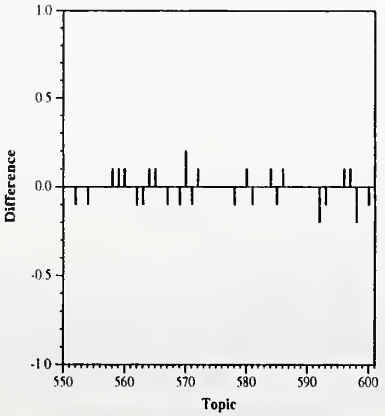
Mean average precision	
non-interpolated	0.0396

Document Level Averages	
	Precision
At 5 docs	0.1184
At 10 docs	0.1082
At 15 docs	0.0898
At 20 docs	0.0776
At 30 docs	0.0728
At 100 docs	0.0429
At 200 docs	0.0293
At 500 docs	0.0230
At 1000 docs	0.0203

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0646



Recall-Precision Curve



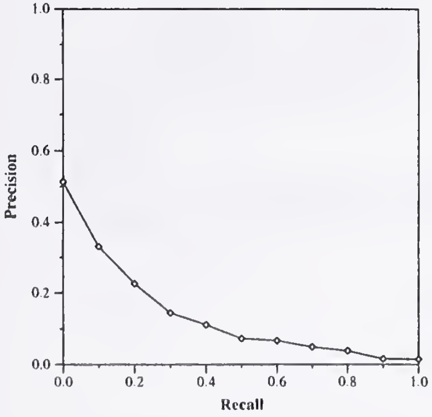
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	thutd1
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

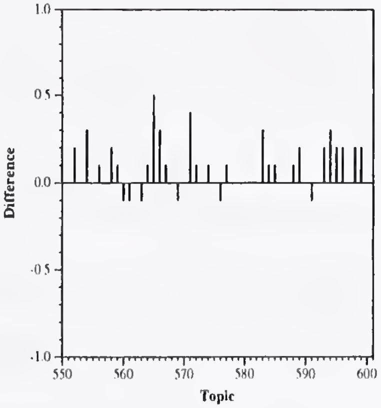
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	1121

Recall Level Averages	
Recall	Precision
0.00	0.5131
0.10	0.3309
0.20	0.2260
0.30	0.1444
0.40	0.1116
0.50	0.0731
0.60	0.0670
0.70	0.0493
0.80	0.0383
0.90	0.0161
1.00	0.0145
Mean average precision	
non-interpolated	0.1225

Document Level Averages	
	Precision
At 5 docs	0.2490
At 10 docs	0.1980
At 15 docs	0.1741
At 20 docs	0.1571
At 30 docs	0.1361
At 100 docs	0.0794
At 200 docs	0.0545
At 500 docs	0.0308
At 1000 docs	0.0229
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1505



Recall-Precision Curve



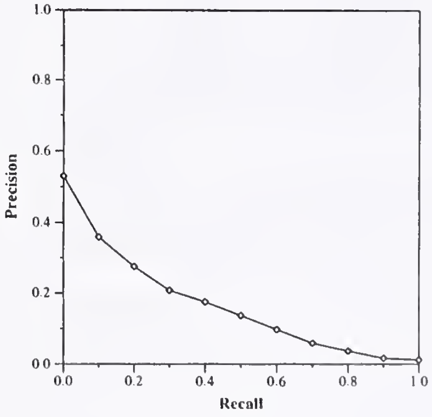
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	thutd2
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

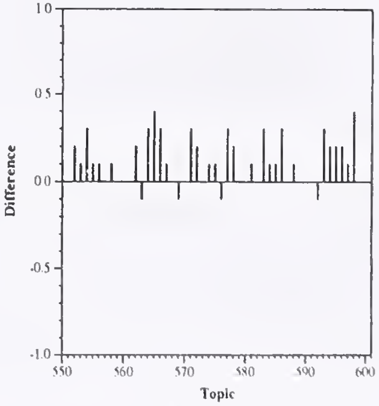
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	1114

Recall Level Averages	
Recall	Precision
0.00	0.5300
0.10	0.3593
0.20	0.2751
0.30	0.2084
0.40	0.1757
0.50	0.1373
0.60	0.0983
0.70	0.0595
0.80	0.0375
0.90	0.0168
1.00	0.0121
Mean average precision	
non-interpolated	0.1555

Document Level Averages	
	Precision
At 5 docs	0.2816
At 10 docs	0.2245
At 15 docs	0.1946
At 20 docs	0.1816
At 30 docs	0.1612
At 100 docs	0.0967
At 200 docs	0.0694
At 500 docs	0.0409
At 1000 docs	0.0227
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1921



Recall-Precision Curve



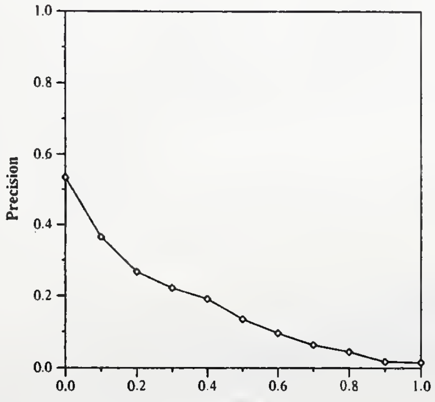
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	thutd3
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

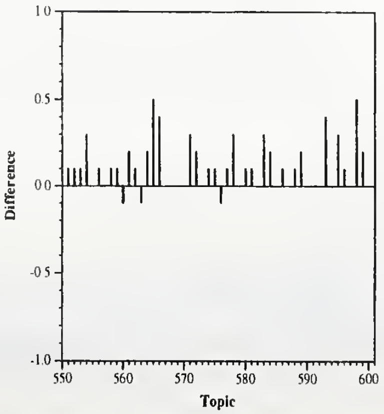
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	1121

Recall Level Averages	
Recall	Precision
0.00	0.5335
0.10	0.3649
0.20	0.2675
0.30	0.2224
0.40	0.1918
0.50	0.1353
0.60	0.0963
0.70	0.0637
0.80	0.0445
0.90	0.0173
1.00	0.0150
Mean average precision	
non-interpolated	0.1577

Document Level Averages	
	Precision
At 5 docs	0.2735
At 10 docs	0.2306
At 15 docs	0.1932
At 20 docs	0.1837
At 30 docs	0.1619
At 100 docs	0.1010
At 200 docs	0.0690
At 500 docs	0.0394
At 1000 docs	0.0229
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1885



Recall-Precision Curve



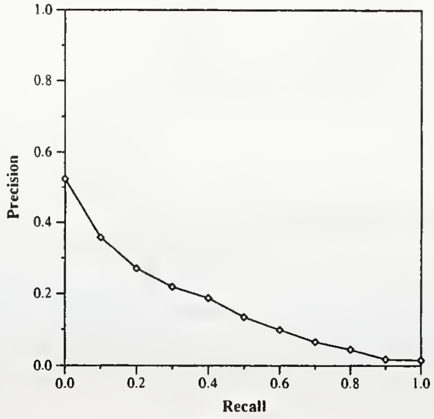
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	thutd4
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

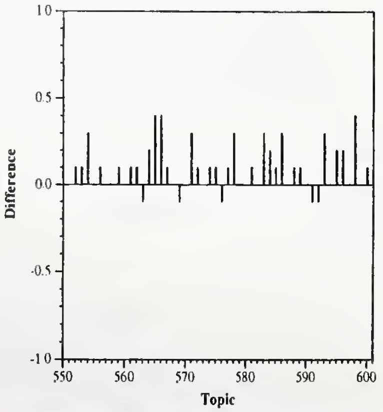
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	1121

Recall Level Averages	
Recall	Precision
0.00	0.5234
0.10	0.3569
0.20	0.2705
0.30	0.2195
0.40	0.1879
0.50	0.1347
0.60	0.0990
0.70	0.0657
0.80	0.0446
0.90	0.0175
1.00	0.0150
Mean average precision	
non-interpolated	0.1567

Document Level Averages	
	Precision
At 5 docs	0.2816
At 10 docs	0.2143
At 15 docs	0.1959
At 20 docs	0.1816
At 30 docs	0.1633
At 100 docs	0.1006
At 200 docs	0.0698
At 500 docs	0.0397
At 1000 docs	0.0229
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1883



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	thutd5
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

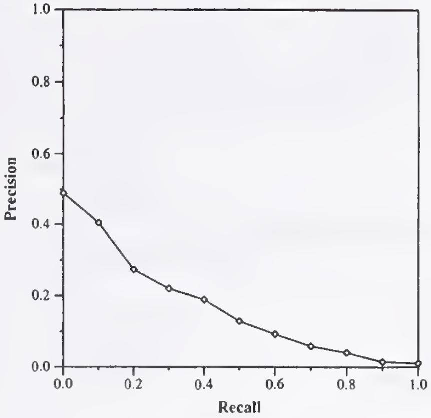
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	1029

Recall Level Averages	
Recall	Precision
0.00	0.4889
0.10	0.4045
0.20	0.2730
0.30	0.2203
0.40	0.1896
0.50	0.1294
0.60	0.0929
0.70	0.0590
0.80	0.0407
0.90	0.0148
1.00	0.0110

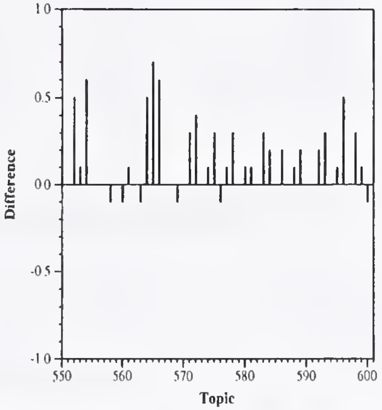
Mean average precision	
non-interpolated	0.1571

Document Level Averages	
	Precision
At 5 docs	0.2694
At 10 docs	0.2510
At 15 docs	0.2218
At 20 docs	0.1959
At 30 docs	0.1619
At 100 docs	0.0976
At 200 docs	0.0682
At 500 docs	0.0367
At 1000 docs	0.0210

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1965



Recall-Precision Curve



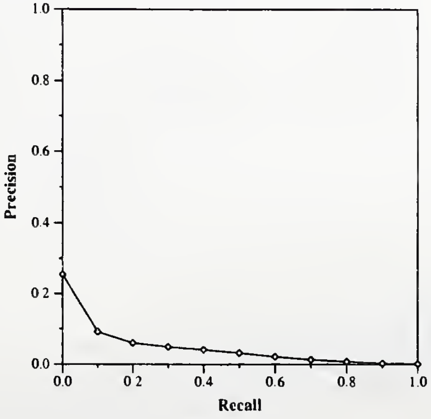
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	pirc2Wd1
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

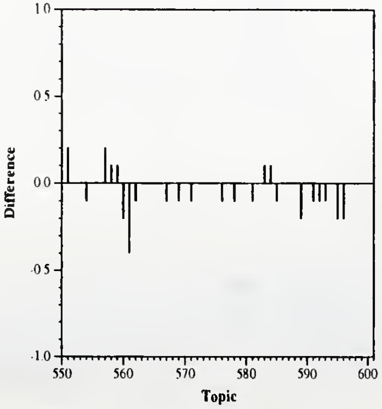
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	839

Recall Level Averages	
Recall	Precision
0.00	0.2539
0.10	0.0915
0.20	0.0604
0.30	0.0492
0.40	0.0416
0.50	0.0324
0.60	0.0217
0.70	0.0132
0.80	0.0075
0.90	0.0029
1.00	0.0014
Mean average precision	
non-interpolated	0.0407

Document Level Averages	
	Precision
At 5 docs	0.1020
At 10 docs	0.0816
At 15 docs	0.0857
At 20 docs	0.0765
At 30 docs	0.0633
At 100 docs	0.0412
At 200 docs	0.0338
At 500 docs	0.0223
At 1000 docs	0.0171
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0538



Recall-Precision Curve



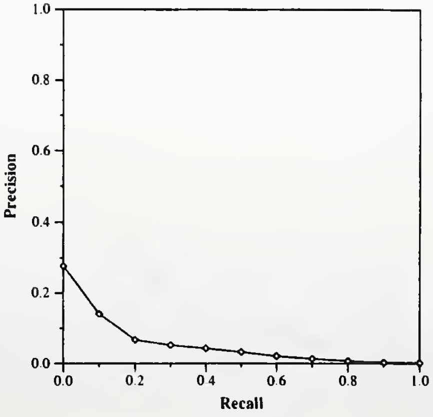
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	pirc2Wd2
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

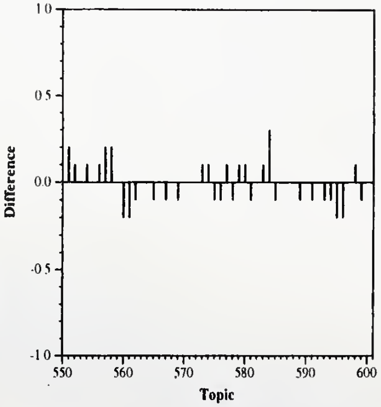
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	840

Recall Level Averages	
Recall	Precision
0.00	0.2768
0.10	0.1409
0.20	0.0670
0.30	0.0523
0.40	0.0431
0.50	0.0327
0.60	0.0213
0.70	0.0133
0.80	0.0073
0.90	0.0027
1.00	0.0013
Mean average precision	
non-interpolated	0.0473

Document Level Averages	
	Precision
At 5 docs	0.1020
At 10 docs	0.1082
At 15 docs	0.1020
At 20 docs	0.0857
At 30 docs	0.0741
At 100 docs	0.0439
At 200 docs	0.0347
At 500 docs	0.0223
At 1000 docs	0.0171
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0775



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Web track, topic distillation task results — University of Amsterdam-Monz

Summary Statistics	
Run ID:	UAmsT02WtA
Run Description	Realistic; NODOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

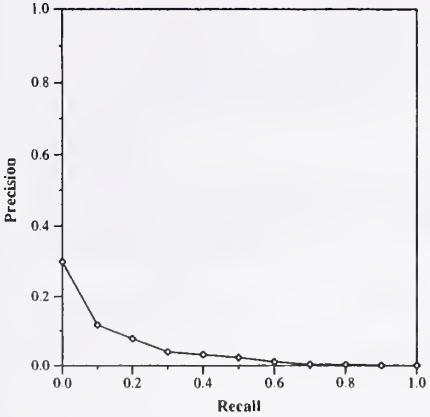
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	354

Recall Level Averages	
Recall	Precision
0.00	0.2999
0.10	0.1179
0.20	0.0775
0.30	0.0391
0.40	0.0312
0.50	0.0231
0.60	0.0105
0.70	0.0032
0.80	0.0026
0.90	0.0000
1.00	0.0000

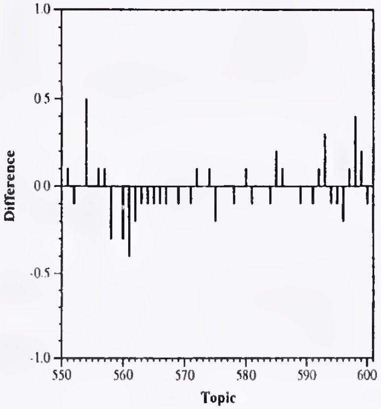
Mean average precision	
non-interpolated	0.0406

Document Level Averages	
	Precision
At 5 docs	0.1224
At 10 docs	0.1000
At 15 docs	0.0803
At 20 docs	0.0714
At 30 docs .	0.0558
At 100 docs	0.0298
At 200 docs	0.0197
At 500 docs	0.0110
At 1000 docs	0.0072

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0632



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	UAmsT02WtAcs
Run Description	Realistic; NODOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

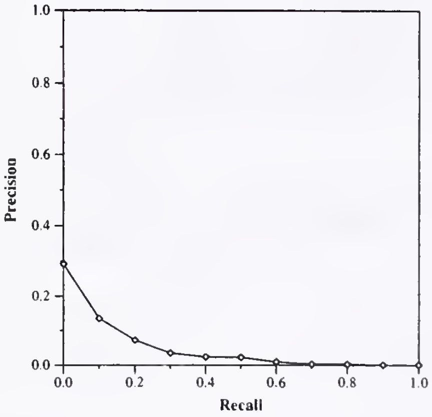
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	354

Recall Level Averages	
Recall	Precision
0.00	0.2924
0.10	0.1351
0.20	0.0718
0.30	0.0343
0.40	0.0233
0.50	0.0218
0.60	0.0092
0.70	0.0029
0.80	0.0023
0.90	0.0000
1.00	0.0000

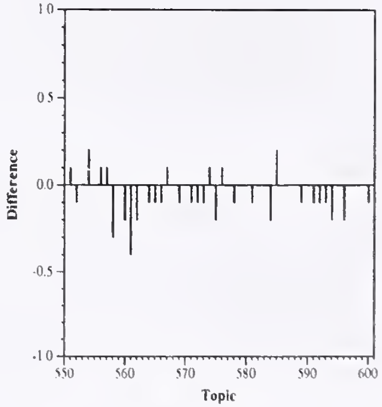
Mean average precision	
non-interpolated	0.0385

Document Level Averages	
	Precision
At 5 docs	0.1061
At 10 docs	0.0653
At 15 docs	0.0748
At 20 docs	0.0786
At 30 docs	0.0660
At 100 docs	0.0341
At 200 docs	0.0218
At 500 docs	0.0118
At 1000 docs	0.0072

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0765



Recall-Precision Curve



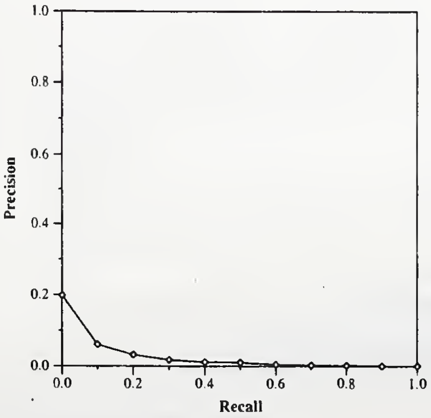
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	UAmsT02WtAri
Run Description	Realistic; NODOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

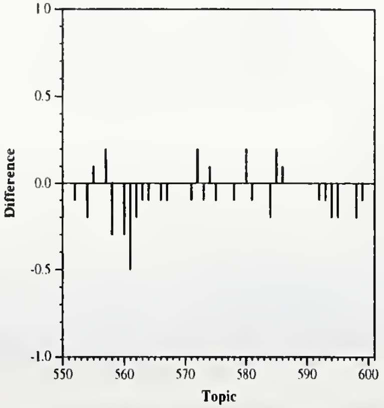
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	387

Recall Level Averages	
Recall	Precision
0.00	0.1981
0.10	0.0609
0.20	0.0320
0.30	0.0175
0.40	0.0114
0.50	0.0104
0.60	0.0045
0.70	0.0024
0.80	0.0015
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0218

Document Level Averages	
	Precision
At 5 docs	0.0735
At 10 docs	0.0633
At 15 docs	0.0503
At 20 docs	0.0469
At 30 docs	0.0381
At 100 docs	0.0163
At 200 docs	0.0149
At 500 docs	0.0115
At 1000 docs	0.0079
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0357



Recall-Precision Curve



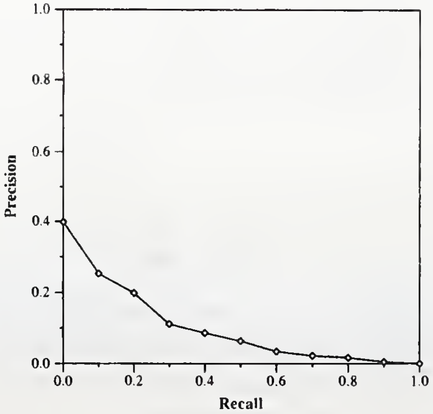
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	UAmsT02WtT
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

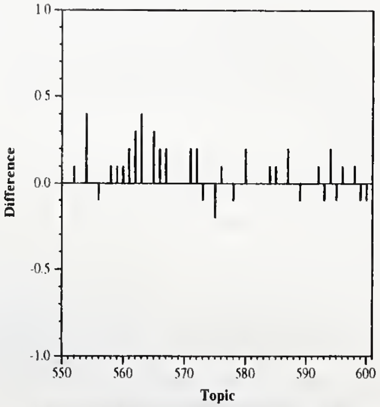
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	591

Recall Level Averages	
Recall	Precision
0.00	0.3986
0.10	0.2528
0.20	0.1984
0.30	0.1117
0.40	0.0867
0.50	0.0642
0.60	0.0347
0.70	0.0223
0.80	0.0169
0.90	0.0050
1.00	0.0003
Mean average precision	
non-interpolated	0.0918

Document Level Averages	
	Precision
At 5 docs	0.1796
At 10 docs	0.1755
At 15 docs	0.1415
At 20 docs	0.1245
At 30 docs	0.1020
At 100 docs	0.0618
At 200 docs	0.0406
At 500 docs	0.0211
At 1000 docs	0.0121
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1140



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

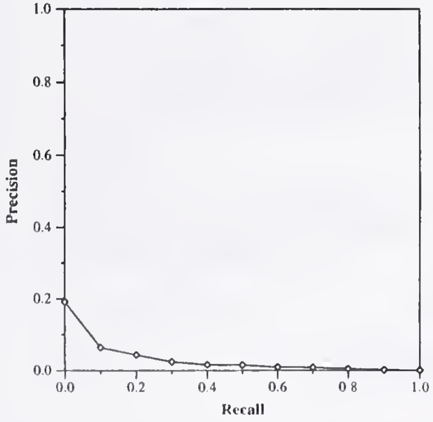
Web track, topic distillation task results — University of Amsterdam-Monz

Summary Statistics	
Run ID:	UAmsT02WtTri
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

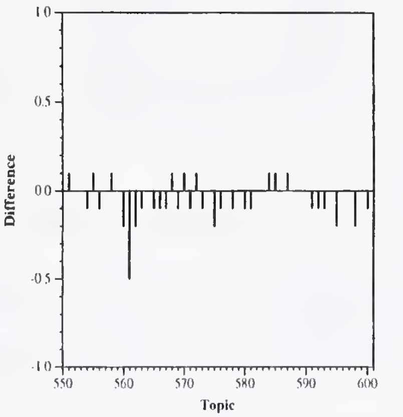
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	621

Recall Level Averages	
Recall	Precision
0.00	0.1919
0.10	0.0639
0.20	0.0434
0.30	0.0244
0.40	0.0163
0.50	0.0154
0.60	0.0095
0.70	0.0085
0.80	0.0044
0.90	0.0015
1.00	0.0003
Mean average precision	
non-interpolated	0.0248

Document Level Averages	
	Precision
At 5 docs	0.0735
At 10 docs	0.0673
At 15 docs	0.0667
At 20 docs	0.0582
At 30 docs	0.0463
At 100 docs	0.0229
At 200 docs	0.0173
At 500 docs	0.0196
At 1000 docs	0.0127
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0470



Recall-Precision Curve



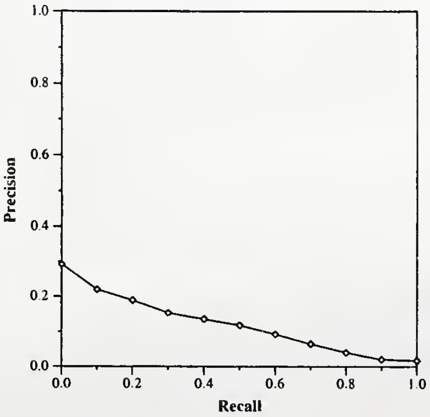
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	uog01ctaialh
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

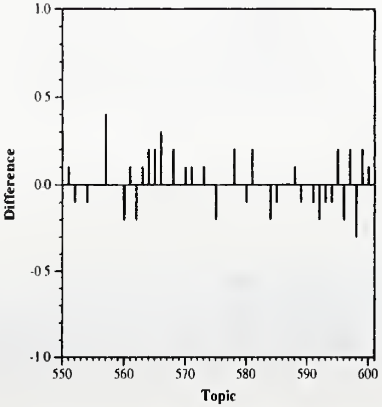
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	1112

Recall Level Averages	
Recall	Precision
0.00	0.2923
0.10	0.2194
0.20	0.1893
0.30	0.1530
0.40	0.1356
0.50	0.1181
0.60	0.0924
0.70	0.0647
0.80	0.0399
0.90	0.0204
1.00	0.0167
Mean average precision	
non-interpolated	0.1072

Document Level Averages	
	Precision
At 5 docs	0.1061
At 10 docs	0.1306
At 15 docs	0.1252
At 20 docs	0.1255
At 30 docs	0.1218
At 100 docs	0.0933
At 200 docs	0.0688
At 500 docs	0.0390
At 1000 docs	0.0227
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1211



Recall-Precision Curve



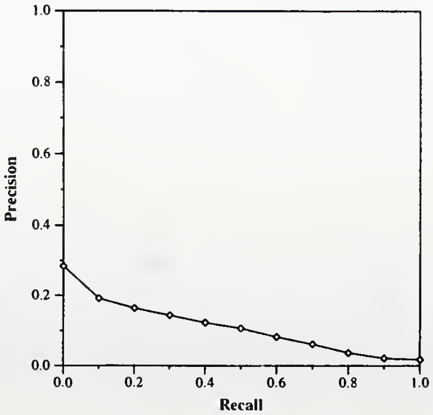
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	uog02ctadh
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

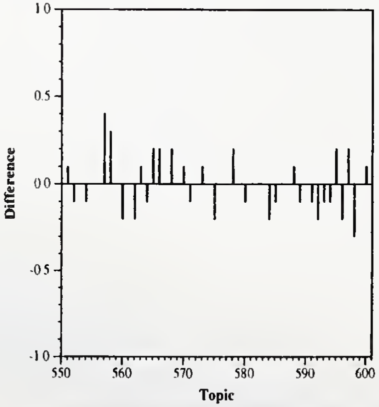
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	1112

Recall Level Averages	
Recall	Precision
0.00	0.2835
0.10	0.1915
0.20	0.1631
0.30	0.1432
0.40	0.1220
0.50	0.1060
0.60	0.0819
0.70	0.0612
0.80	0.0366
0.90	0.0208
1.00	0.0170
Mean average precision	
non-interpolated	0.0979

Document Level Averages	
	Precision
At 5 docs	0.0980
At 10 docs	0.1143
At 15 docs	0.1116
At 20 docs	0.1184
At 30 docs	0.1116
At 100 docs	0.0933
At 200 docs	0.0691
At 500 docs	0.0390
At 1000 docs	0.0227
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1095



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	uog03ctadqh
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

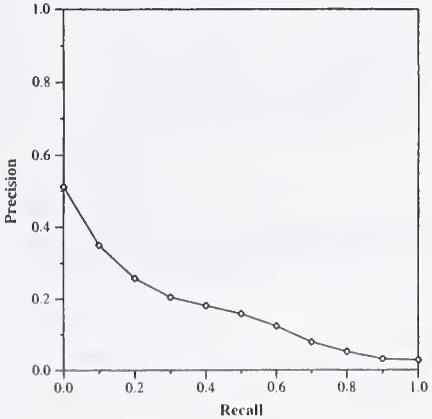
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	1112

Recall Level Averages	
Recall	Precision
0.00	0.5117
0.10	0.3499
0.20	0.2575
0.30	0.2048
0.40	0.1810
0.50	0.1581
0.60	0.1237
0.70	0.0785
0.80	0.0512
0.90	0.0316
1.00	0.0279

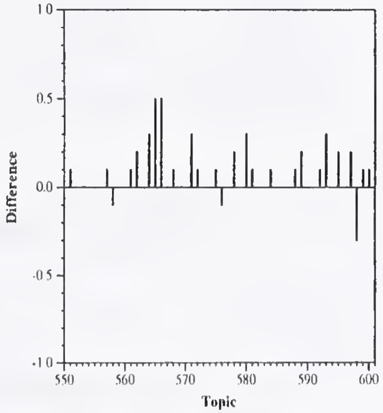
Mean average precision	
non-interpolated	0.1582

Document Level Averages	
	Precision
At 5 docs	0.2367
At 10 docs	0.1939
At 15 docs	0.1782
At 20 docs	0.1612
At 30 docs	0.1476
At 100 docs	0.0916
At 200 docs	0.0689
At 500 docs	0.0388
At 1000 docs	0.0227

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1918



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	uog04cta2dqh
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

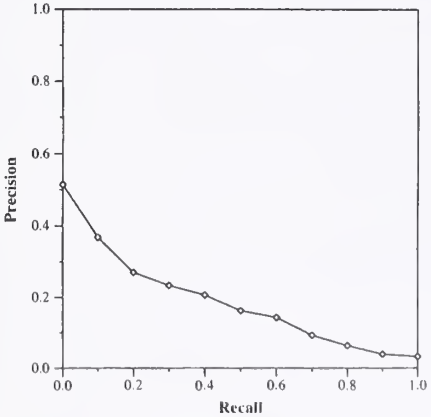
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	1163

Recall Level Averages	
Recall	Precision
0.00	0.5142
0.10	0.3685
0.20	0.2703
0.30	0.2339
0.40	0.2067
0.50	0.1628
0.60	0.1441
0.70	0.0935
0.80	0.0638
0.90	0.0397
1.00	0.0332

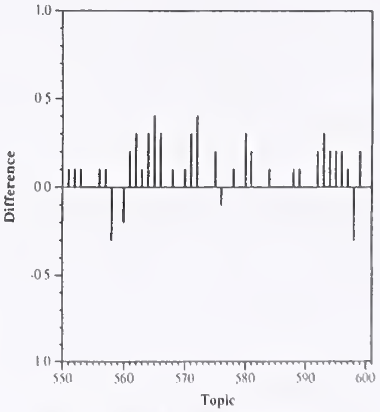
Mean average precision	
non-interpolated	0.1743

Document Level Averages	
	Precision
At 5 docs	0.2286
At 10 docs	0.2082
At 15 docs	0.1891
At 20 docs	0.1704
At 30 docs	0.1469
At 100 docs	0.1041
At 200 docs	0.0755
At 500 docs	0.0416
At 1000 docs	0.0237

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1980



Recall-Precision Curve



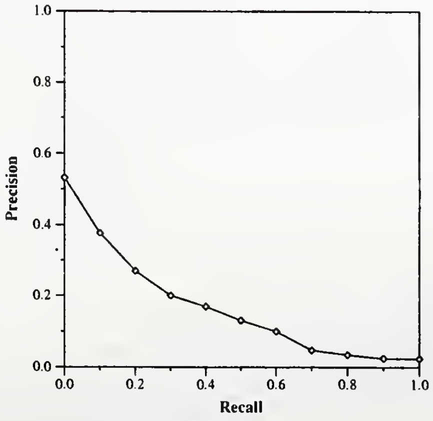
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	uog05tad
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

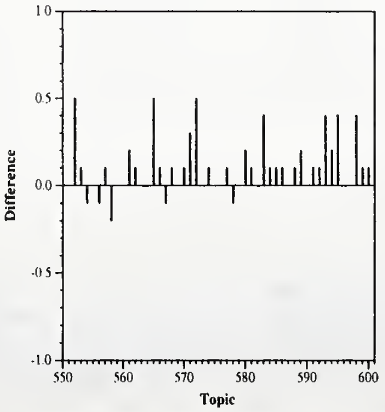
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	757

Recall Level Averages	
Recall	Precision
0.00	0 5313
0.10	0.3768
0.20	0.2698
0.30	0.1997
0.40	0.1689
0.50	0.1307
0.60	0.0999
0.70	0.0476
0.80	0.0344
0.90	0.0245
1.00	0.0236
Mean average precision	
non-interpolated	0.1540

Document Level Averages	
	Precision
At 5 docs	0.2776
At 10 docs	0.2224
At 15 docs	0.2014
At 20 docs	0.1765
At 30 docs	0.1565
At 100 docs	0.0886
At 200 docs	0.0559
At 500 docs	0.0276
At 1000 docs	0.0154
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1930



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

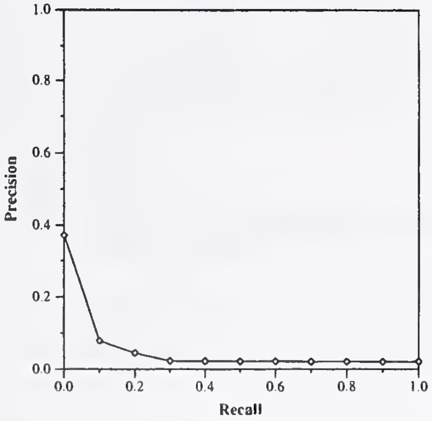
Web track, topic distillation task results — University of Illinois at Chicago

Summary Statistics	
Run ID:	uic0101
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

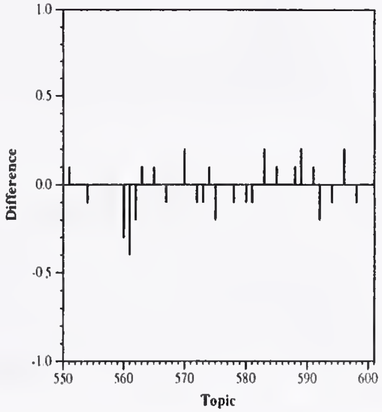
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	60

Recall Level Averages	
Recall	Precision
0.00	0.3721
0.10	0.0787
0.20	0.0446
0.30	0.0228
0.40	0.0221
0.50	0.0217
0.60	0.0217
0.70	0.0208
0.80	0.0207
0.90	0.0204
1.00	0.0204
Mean average precision	
non-interpolated	0.0459

Document Level Averages	
	Precision
At 5 docs	0.1469
At 10 docs	0.1000
At 15 docs	0.0680
At 20 docs	0.0520
At 30 docs	0.0347
At 100 docs	0.0112
At 200 docs	0.0059
At 500 docs	0.0024
At 1000 docs	0.0012
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0593



Recall-Precision Curve



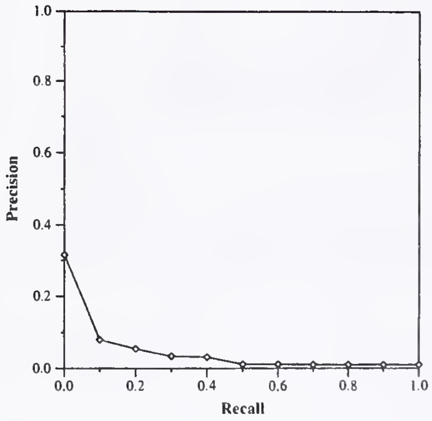
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	uic0102
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

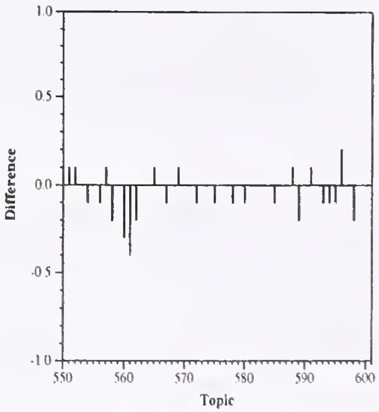
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	50

Recall Level Averages	
Recall	Precision
0.00	0.3156
0.10	0.0804
0.20	0.0549
0.30	0.0345
0.40	0.0322
0.50	0.0116
0.60	0.0116
0.70	0.0106
0.80	0.0106
0.90	0.0102
1.00	0.0102
Mean average precision	
non-interpolated	0.0375

Document Level Averages	
	Precision
At 5 docs	0.1102
At 10 docs	0.0796
At 15 docs	0.0531
At 20 docs	0.0418
At 30 docs	0.0293
At 100 docs	0.0092
At 200 docs	0.0049
At 500 docs	0.0020
At 1000 docs	0.0010
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0390



Recall-Precision Curve



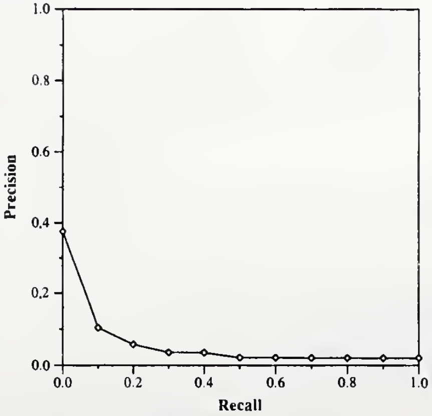
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	uic0103
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

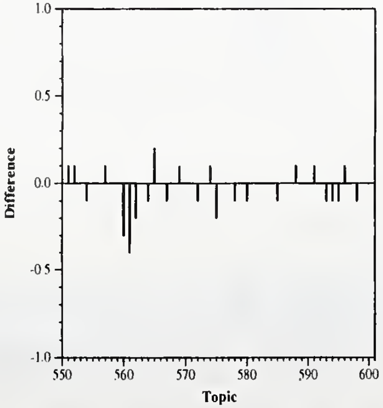
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	55

Recall Level Averages	
Recall	Precision
0.00	0.3759
0.10	0.1042
0.20	0.0582
0.30	0.0364
0.40	0.0357
0.50	0.0217
0.60	0.0217
0.70	0.0208
0.80	0.0207
0.90	0.0204
1.00	0.0204
Mean average precision	
non-interpolated	0.0489

Document Level Averages	
	Precision
At 5 docs	0.1224
At 10 docs	0.0898
At 15 docs	0.0612
At 20 docs	0.0469
At 30 docs	0.0313
At 100 docs	0.0102
At 200 docs	0.0054
At 500 docs	0.0022
At 1000 docs	0.0011
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0643



Recall-Precision Curve



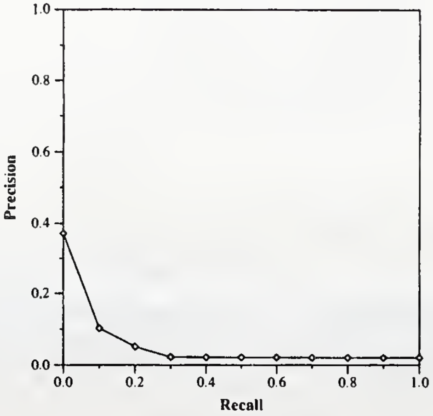
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	uic0104
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

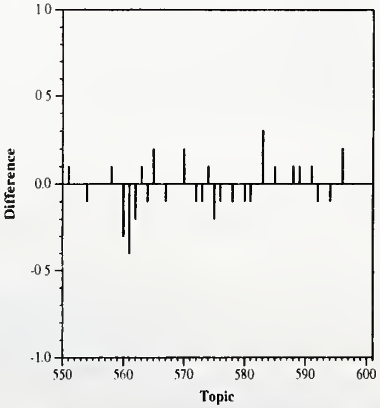
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	63

Recall Level Averages	
Recall	Precision
0.00	0.3713
0.10	0.1025
0.20	0.0514
0.30	0.0227
0.40	0.0221
0.50	0.0217
0.60	0.0217
0.70	0.0208
0.80	0.0207
0.90	0.0204
1.00	0.0204
Mean average precision	
non-interpolated	0.0486

Document Level Averages	
	Precision
At 5 docs	0.1510
At 10 docs	0.1041
At 15 docs	0.0707
At 20 docs	0.0541
At 30 docs	0.0361
At 100 docs	0.0116
At 200 docs	0.0061
At 500 docs	0.0025
At 1000 docs	0.0013
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0635



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

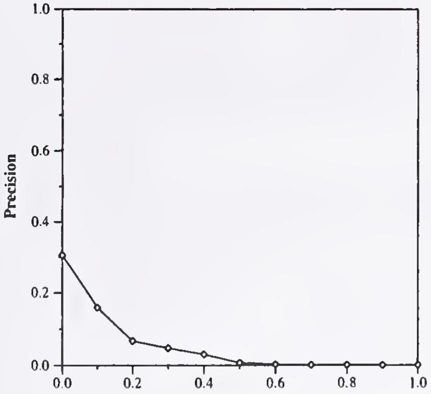
Web track, topic distillation task results — University of Maryland Baltimore County-Cost

Summary Statistics	
Run ID:	CARROT2A
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

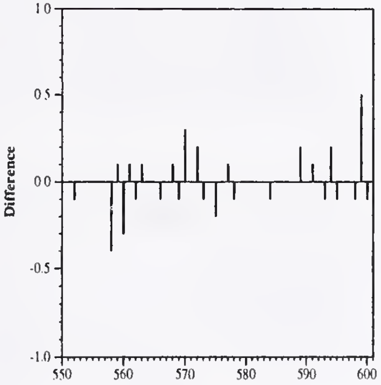
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	416

Recall Level Averages	
Recall	Precision
0.00	0.3044
0.10	0.1602
0.20	0.0674
0.30	0.0481
0.40	0.0305
0.50	0.0070
0.60	0.0019
0.70	0.0005
0.80	0.0003
0.90	0.0001
1.00	0.0001
Mean average precision	
non-interpolated	0.0430

Document Level Averages	
	Precision
At 5 docs	0.1510
At 10 docs	0.1143
At 15 docs	0.1007
At 20 docs	0.0827
At 30 docs	0.0680
At 100 docs	0.0369
At 200 docs	0.0243
At 500 docs	0.0138
At 1000 docs	0.0085
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0784



Recall-Precision Curve



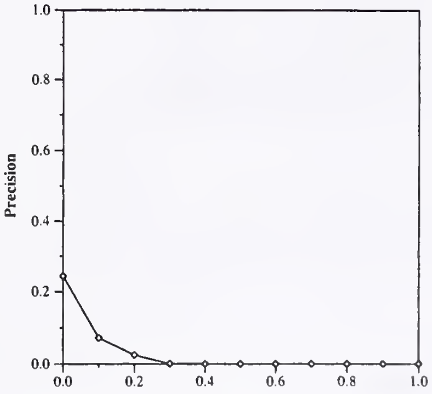
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	CARROT2B
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

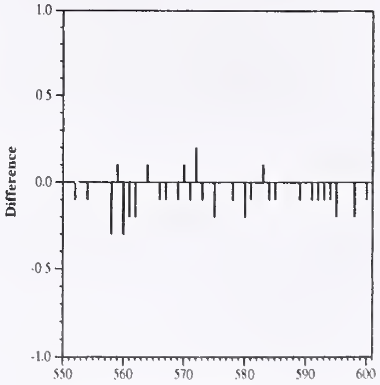
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	165

Recall Level Averages	
Recall	Precision
0.00	0.2437
0.10	0.0724
0.20	0.0249
0.30	0.0009
0.40	0.0000
0.50	0.0000
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0187

Document Level Averages	
	Precision
At 5 docs	0.0857
At 10 docs	0.0551
At 15 docs	0.0517
At 20 docs	0.0439
At 30 docs	0.0361
At 100 docs	0.0196
At 200 docs	0.0115
At 500 docs	0.0056
At 1000 docs	0.0034
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0346



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	CARROT2C
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

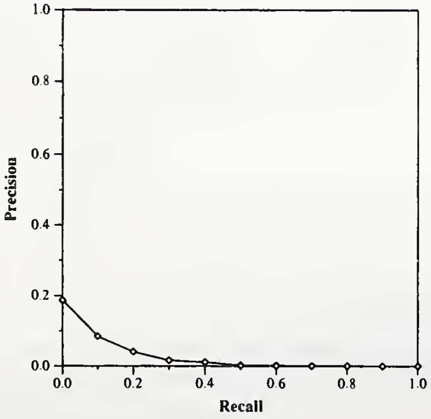
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	396

Recall Level Averages	
Recall	Precision
0.00	0.1875
0.10	0.0847
0.20	0.0409
0.30	0.0163
0.40	0.0112
0.50	0.0020
0.60	0.0012
0.70	0.0005
0.80	0.0002
0.90	0.0000
1.00	0.0000

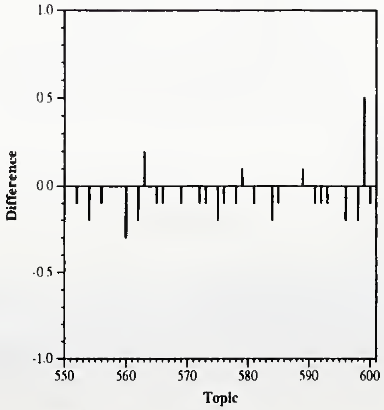
Mean average precision	
non-interpolated	0.0229

Document Level Averages	
	Precision
At 5 docs	0.0816
At 10 docs	0.0714
At 15 docs	0.0531
At 20 docs	0.0449
At 30 docs	0.0415
At 100 docs	0.0245
At 200 docs	0.0176
At 500 docs	0.0109
At 1000 docs	0.0081

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0439



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	CARROT2D
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

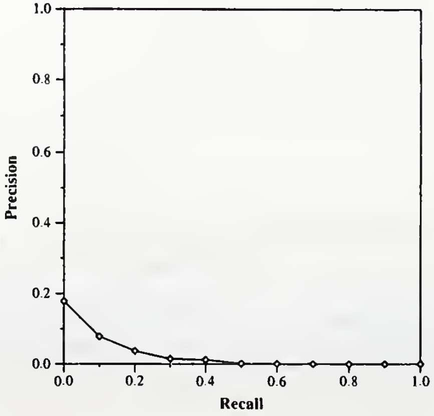
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	397

Recall Level Averages	
Recall	Precision
0.00	0.1787
0.10	0.0785
0.20	0.0371
0.30	0.0149
0.40	0.0118
0.50	0.0016
0.60	0.0009
0.70	0.0004
0.80	0.0002
0.90	0.0000
1.00	0.0000

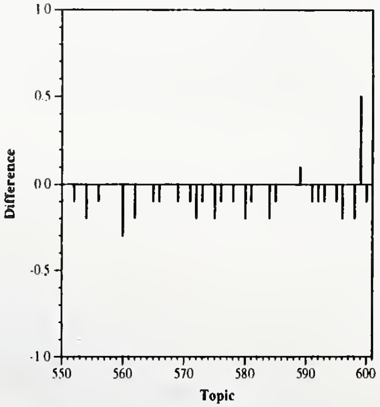
Mean average precision	
non-interpolated	0.0210

Document Level Averages	
	Precision
At 5 docs	0.0776
At 10 docs	0.0551
At 15 docs	0.0490
At 20 docs	0.0429
At 30 docs	0.0381
At 100 docs	0.0224
At 200 docs	0.0165
At 500 docs	0.0109
At 1000 docs	0.0081

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0335



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

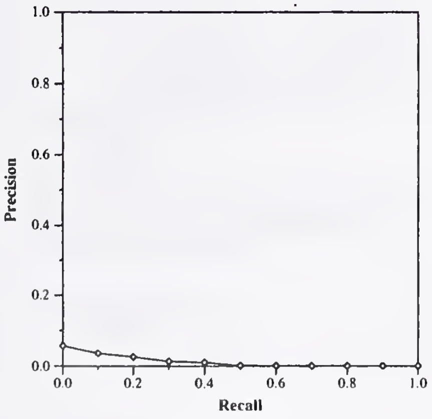
Web track, topic distillation task results — University of Maryland Baltimore County-Cost

Summary Statistics	
Run ID:	CARROT2E
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Number of Topics:	49

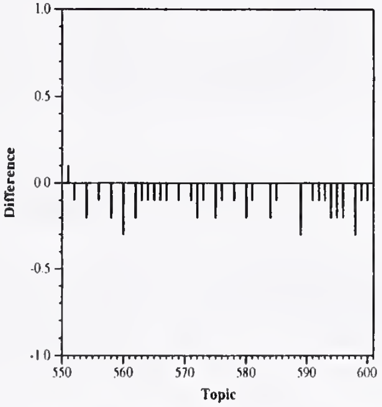
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	332

Recall Level Averages	
Recall	Precision
0.00	0.0572
0.10	0.0364
0.20	0.0261
0.30	0.0134
0.40	0.0103
0.50	0.0014
0.60	0.0006
0.70	0.0002
0.80	0.0001
0.90	0.0001
1.00	0.0001
Mean average precision	
non-interpolated	0.0107

Document Level Averages	
	Precision
At 5 docs	0.0163
At 10 docs	0.0184
At 15 docs	0.0190
At 20 docs	0.0173
At 30 docs	0.0197
At 100 docs	0.0147
At 200 docs	0.0114
At 500 docs	0.0094
At 1000 docs	0.0068
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0160



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	MU111
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

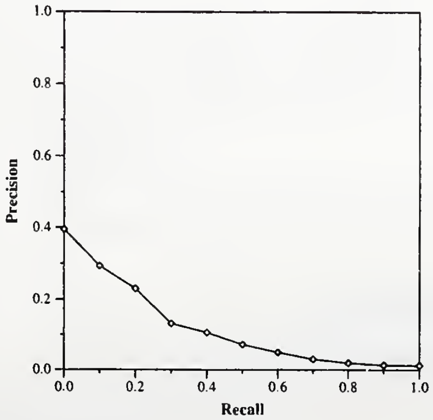
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	918

Recall Level Averages	
Recall	Precision
0.00	0.3950
0.10	0.2937
0.20	0.2309
0.30	0.1322
0.40	0.1066
0.50	0.0723
0.60	0.0497
0.70	0.0300
0.80	0.0193
0.90	0.0134
1.00	0.0122

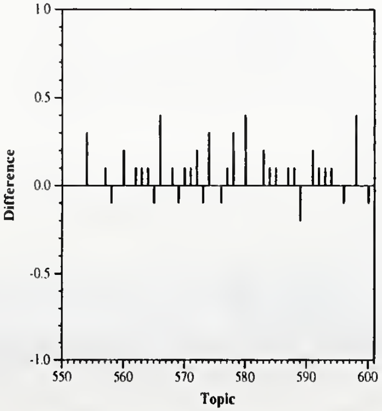
Mean average precision	
non-interpolated	0.1065

Document Level Averages	
	Precision
At 5 docs	0.2000
At 10 docs	0.1857
At 15 docs	0.1646
At 20 docs	0.1500
At 30 docs	0.1204
At 100 docs	0.0680
At 200 docs	0.0490
At 500 docs	0.0294
At 1000 docs	0.0187

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1455



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	MU212
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

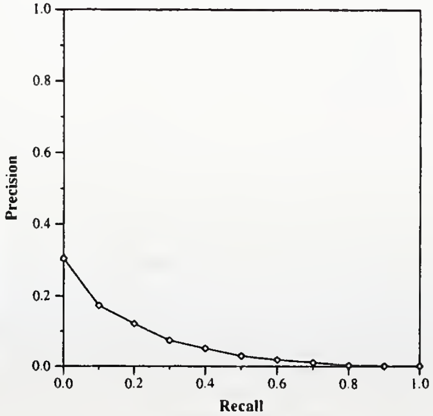
Total number of documents over all topics	
Retrieved:	48658
Relevant:	1574
Rel-ret:	591

Recall Level Averages	
Recall	Precision
0.00	0.3045
0.10	0.1732
0.20	0.1217
0.30	0.0744
0.40	0.0513
0.50	0.0299
0.60	0.0188
0.70	0.0111
0.80	0.0025
0.90	0.0007
1.00	0.0007

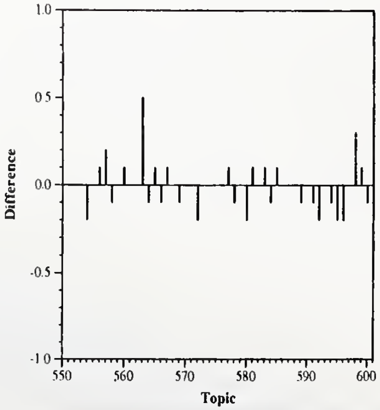
Mean average precision	
non-interpolated	0.0584

Document Level Averages	
	Precision
At 5 docs	0.1184
At 10 docs	0.1082
At 15 docs	0.0980
At 20 docs	0.0867
At 30 docs	0.0789
At 100 docs	0.0482
At 200 docs	0.0336
At 500 docs	0.0195
At 1000 docs	0.0121

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0929



Recall-Precision Curve



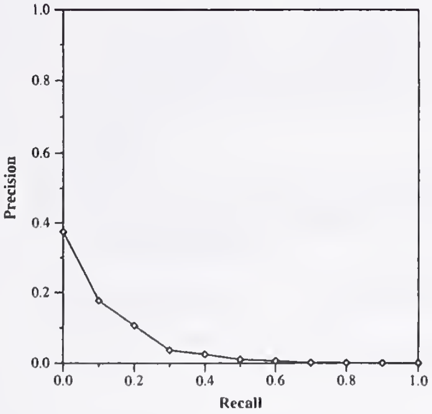
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	MU313
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

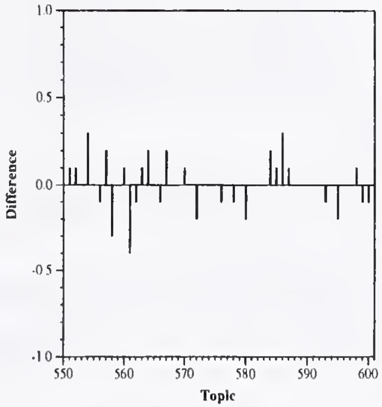
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	400

Recall Level Averages	
Recall	Precision
0.00	0.3742
0.10	0.1772
0.20	0.1068
0.30	0.0367
0.40	0.0253
0.50	0.0108
0.60	0.0065
0.70	0.0018
0.80	0.0009
0.90	0.0006
1.00	0.0006
Mean average precision	
non-interpolated	0.0505

Document Level Averages	
	Precision
At 5 docs	0.1469
At 10 docs	0.1163
At 15 docs	0.0980
At 20 docs	0.0898
At 30 docs	0.0741
At 100 docs	0.0410
At 200 docs	0.0263
At 500 docs	0.0134
At 1000 docs	0.0082
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0931



Recall-Precision Curve



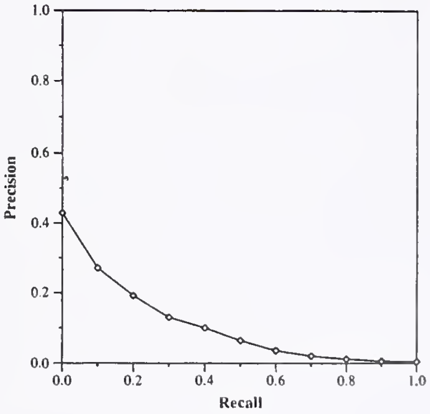
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	MU525
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

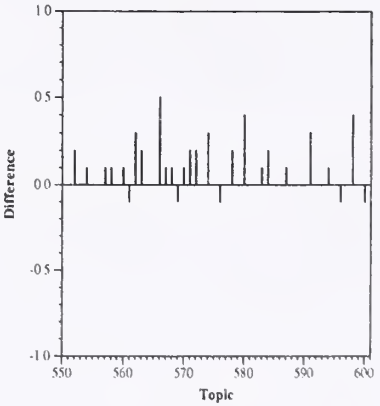
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	938

Recall Level Averages	
Recall	Precision
0.00	0.4285
0.10	0.2705
0.20	0.1914
0.30	0.1296
0.40	0.1000
0.50	0.0644
0.60	0.0358
0.70	0.0206
0.80	0.0121
0.90	0.0059
1.00	0.0046
Mean average precision	
non-interpolated	0.0965

Document Level Averages	
	Precision
At 5 docs	0.2000
At 10 docs	0.1939
At 15 docs	0.1741
At 20 docs	0.1541
At 30 docs	0.1293
At 100 docs	0.0720
At 200 docs	0.0477
At 500 docs	0.0291
At 1000 docs	0.0191
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1463



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Web track, topic distillation task results — The University of Melbourne

Summary Statistics	
Run ID:	MU624
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

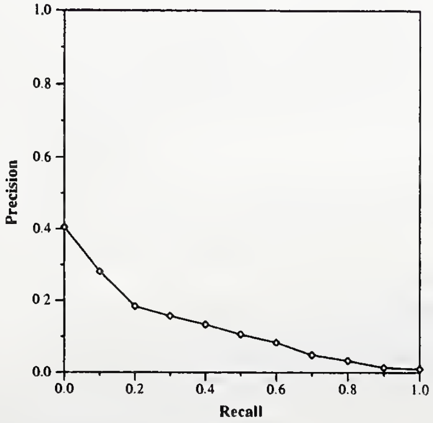
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	1096

Recall Level Averages	
Recall	Precision
0.00	0.4045
0.10	0.2811
0.20	0.1840
0.30	0.1570
0.40	0.1337
0.50	0.1062
0.60	0.0837
0.70	0.0489
0.80	0.0332
0.90	0.0140
1.00	0.0097

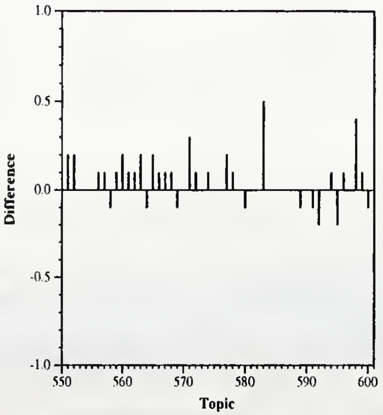
Mean average precision	
non-interpolated	0.1140

Document Level Averages	
	Precision
At 5 docs	0.1878
At 10 docs	0.1694
At 15 docs	0.1497
At 20 docs	0.1388
At 30 docs	0.1197
At 100 docs	0.0892
At 200 docs	0.0654
At 500 docs	0.0374
At 1000 docs	0.0224

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1527



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Web track, topic distillation task results — University of Neuchatel

Summary Statistics	
Run ID:	UniNEdi1
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

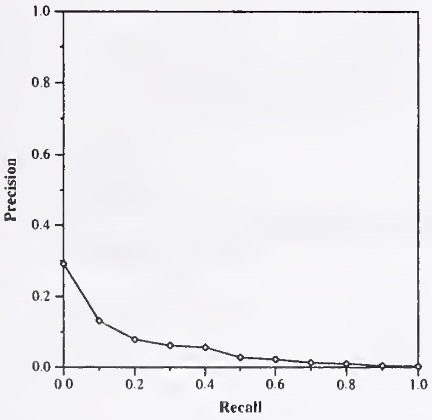
Total number of documents over all topics	
Retrieved:	47432
Relevant:	1574
Rel-ret:	905

Recall Level Averages	
Recall	Precision
0.00	0.2914
0.10	0.1317
0.20	0.0787
0.30	0.0617
0.40	0.0565
0.50	0.0283
0.60	0.0223
0.70	0.0130
0.80	0.0099
0.90	0.0044
1.00	0.0025

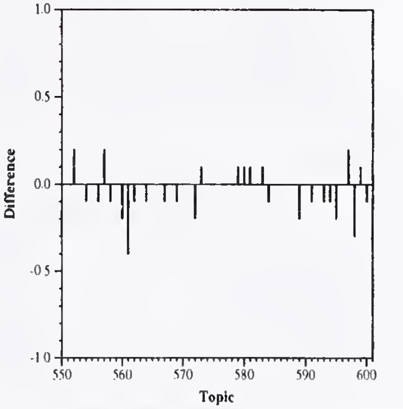
Mean average precision	
non-interpolated	0.0486

Document Level Averages	
	Precision
At 5 docs	0.1184
At 10 docs	0.0837
At 15 docs	0.0667
At 20 docs	0.0561
At 30 docs	0.0469
At 100 docs	0.0390
At 200 docs	0.0287
At 500 docs	0.0241
At 1000 docs	0.0185

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0667



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	UniNEdi2
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

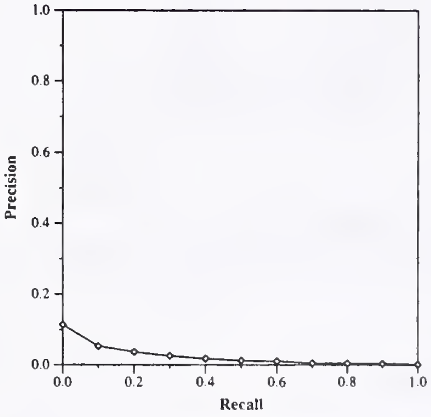
Total number of documents over all topics	
Retrieved:	40517
Relevant:	1574
Rel-ret:	664

Recall Level Averages	
Recall	Precision
0.00	0.1141
0.10	0.0531
0.20	0.0368
0.30	0.0257
0.40	0.0175
0.50	0.0121
0.60	0.0103
0.70	0.0045
0.80	0.0040
0.90	0.0031
1.00	0.0005

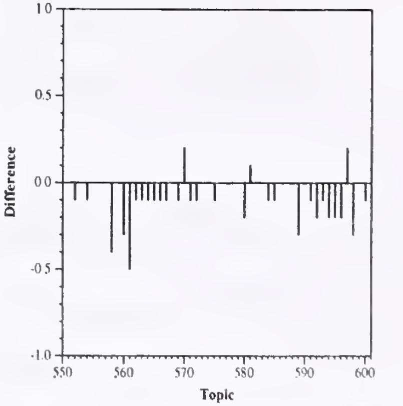
Mean average precision	
non-interpolated	0.0191

Document Level Averages	
	Precision
At 5 docs	0.0327
At 10 docs	0.0327
At 15 docs	0.0286
At 20 docs	0.0224
At 30 docs	0.0231
At 100 docs	0.0218
At 200 docs	0.0190
At 500 docs	0.0180
At 1000 docs	0.0136

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0320



Recall-Precision Curve



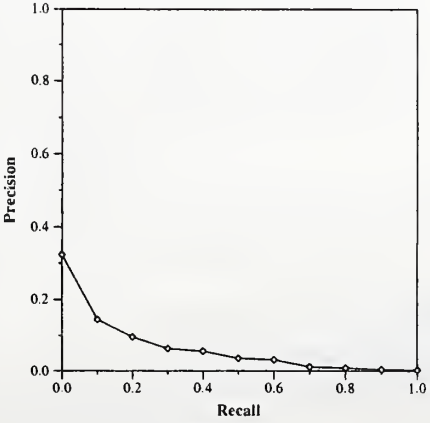
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	UniNEdi3
Run Description	Exploratory; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics:	49

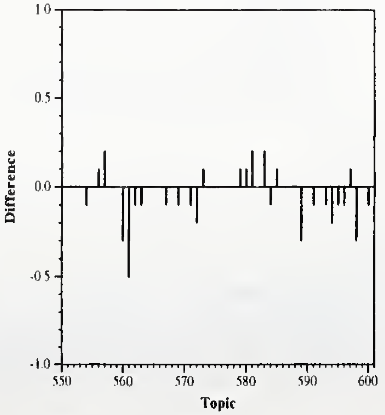
Total number of documents over all topics	
Retrieved:	48996
Relevant:	1574
Rel-ret:	871

Recall Level Averages	
Recall	Precision
0.00	0.3232
0.10	0.1447
0.20	0.0961
0.30	0.0632
0.40	0.0558
0.50	0.0358
0.60	0.0315
0.70	0.0113
0.80	0.0080
0.90	0.0038
1.00	0.0023
Mean average precision	
non-interpolated	0.0547

Document Level Averages	
	Precision
At 5 docs	0.1143
At 10 docs	0.0776
At 15 docs	0.0680
At 20 docs	0.0684
At 30 docs	0.0605
At 100 docs	0.0410
At 200 docs	0.0304
At 500 docs	0.0206
At 1000 docs	0.0178
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0776



Recall-Precision Curve



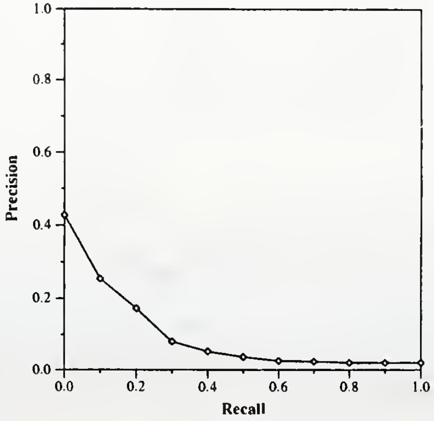
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	UniNEdi4
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

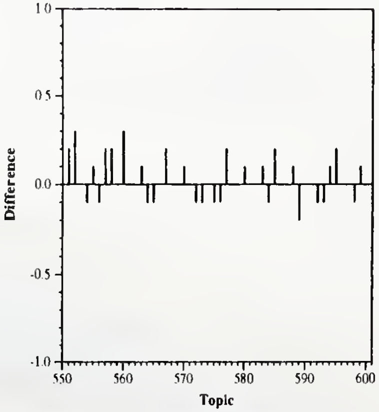
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	457

Recall Level Averages	
Recall	Precision
0.00	0.4274
0.10	0.2546
0.20	0.1722
0.30	0.0808
0.40	0.0528
0.50	0.0372
0.60	0.0259
0.70	0.0240
0.80	0.0206
0.90	0.0204
1.00	0.0204
Mean average precision	
non-interpolated	0.0839

Document Level Averages	
	Precision
At 5 docs	0.1837
At 10 docs	0.1429
At 15 docs	0.1429
At 20 docs	0.1255
At 30 docs	0.1034
At 100 docs	0.0476
At 200 docs	0.0288
At 500 docs	0.0140
At 1000 docs	0.0093
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1341



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	UniNEdi5
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

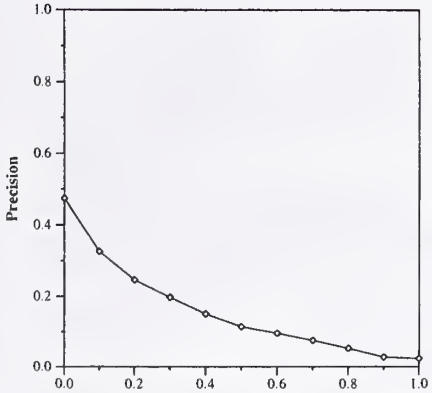
Total number of documents over all topics	
Retrieved:	49000
Relevant:	1574
Rel-ret:	973

Recall Level Averages	
Recall	Precision
0.00	0.4737
0.10	0.3262
0.20	0.2461
0.30	0.1975
0.40	0.1510
0.50	0.1152
0.60	0.0960
0.70	0.0756
0.80	0.0527
0.90	0.0277
1.00	0.0237

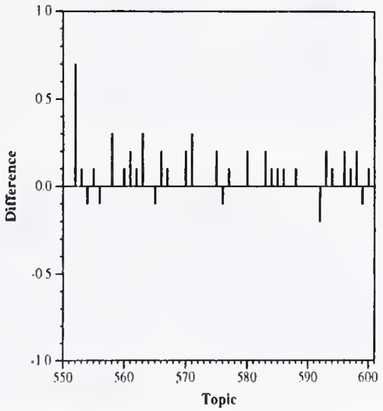
Mean average precision	
non-interpolated	0.1446

Document Level Averages	
	Precision
At 5 docs	0.2408
At 10 docs	0.1959
At 15 docs	0.1864
At 20 docs	0.1724
At 30 docs	0.1490
At 100 docs	0.0898
At 200 docs	0.0628
At 500 docs	0.0340
At 1000 docs	0.0199

R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.1850



Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

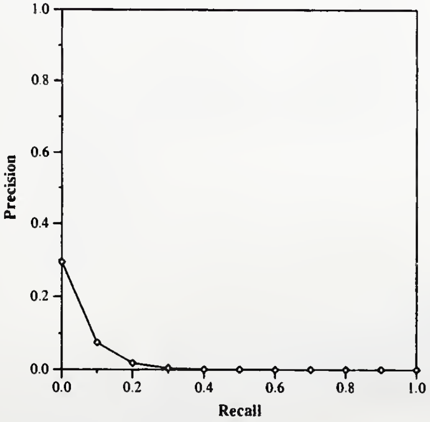
Web track, topic distillation task results — The Univ. of Sunderland-stokoe

Summary Statistics	
Run ID:	TDtfidf
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

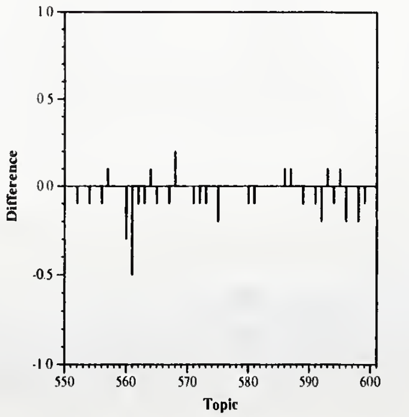
Total number of documents over all topics	
Retrieved:	16251
Relevant:	1574
Rel-ret:	174

Recall Level Averages	
Recall	Precision
0.00	0.2941
0.10	0.0751
0.20	0.0180
0.30	0.0040
0.40	0.0008
0.50	0.0008
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0211

Document Level Averages	
	Precision
At 5 docs	0.0816
At 10 docs	0.0653
At 15 docs	0.0680
At 20 docs	0.0612
At 30 docs	0.0497
At 100 docs	0.0259
At 200 docs	0.0151
At 500 docs	0.0069
At 1000 docs	0.0036
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0451



Recall-Precision Curve



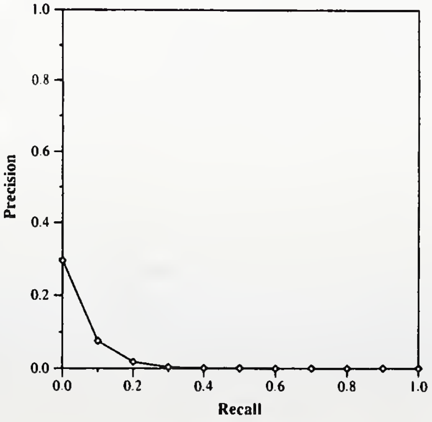
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID:	TDwsdtfidf
Run Description	Realistic; NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Number of Topics:	49

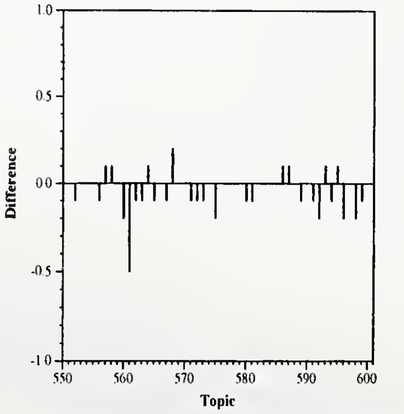
Total number of documents over all topics	
Retrieved:	16251
Relevant:	1574
Rel-ret:	174

Recall Level Averages	
Recall	Precision
0.00	0.2952
0.10	0.0760
0.20	0.0181
0.30	0.0038
0.40	0.0008
0.50	0.0008
0.60	0.0000
0.70	0.0000
0.80	0.0000
0.90	0.0000
1.00	0.0000
Mean average precision	
non-interpolated	0.0211

Document Level Averages	
	Precision
At 5 docs	0.0857
At 10 docs	0.0714
At 15 docs	0.0680
At 20 docs	0.0633
At 30 docs	0.0497
At 100 docs	0.0251
At 200 docs	0.0148
At 500 docs	0.0069
At 1000 docs	0.0036
R-Precision: precision after R (number relevant) documents retrieved	
Exact	0.0454

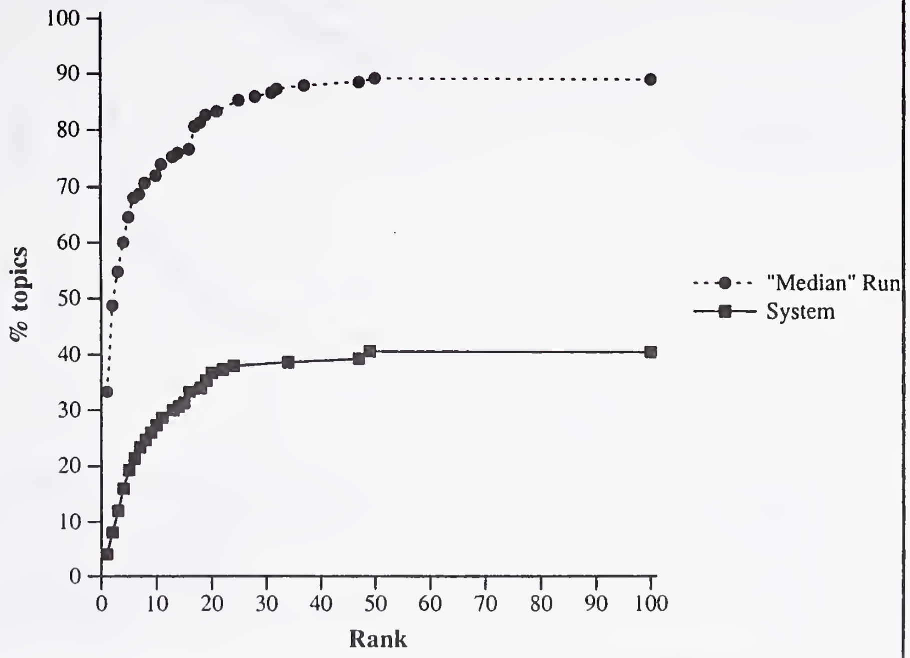


Recall-Precision Curve



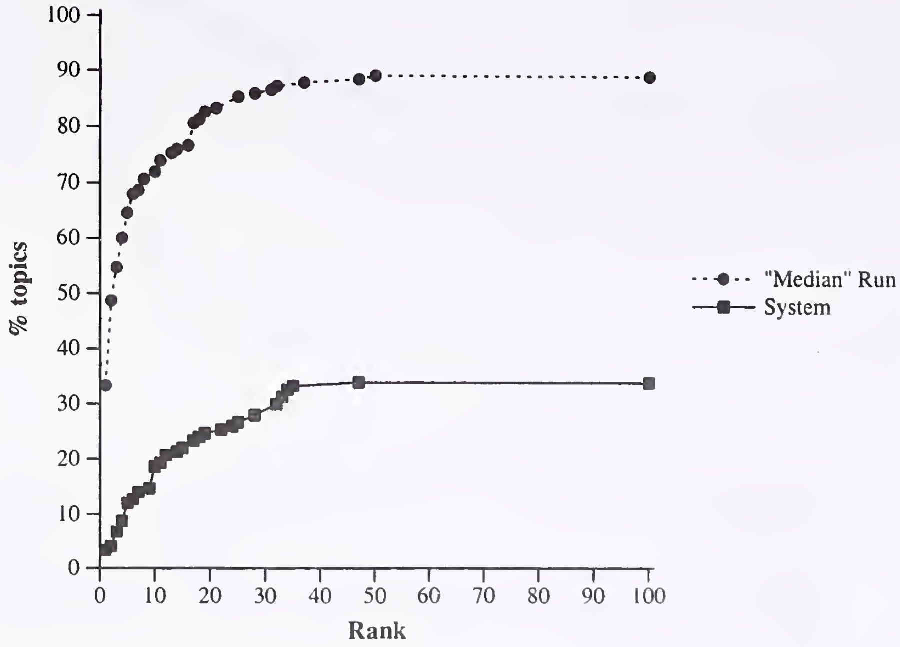
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID	ajouai0201
Run Description	DOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.108
Num found at rank 1	6 (4.0%)
Num found in top 10	41 (27.3%)
Num not found in top 100	89 (59.3%)



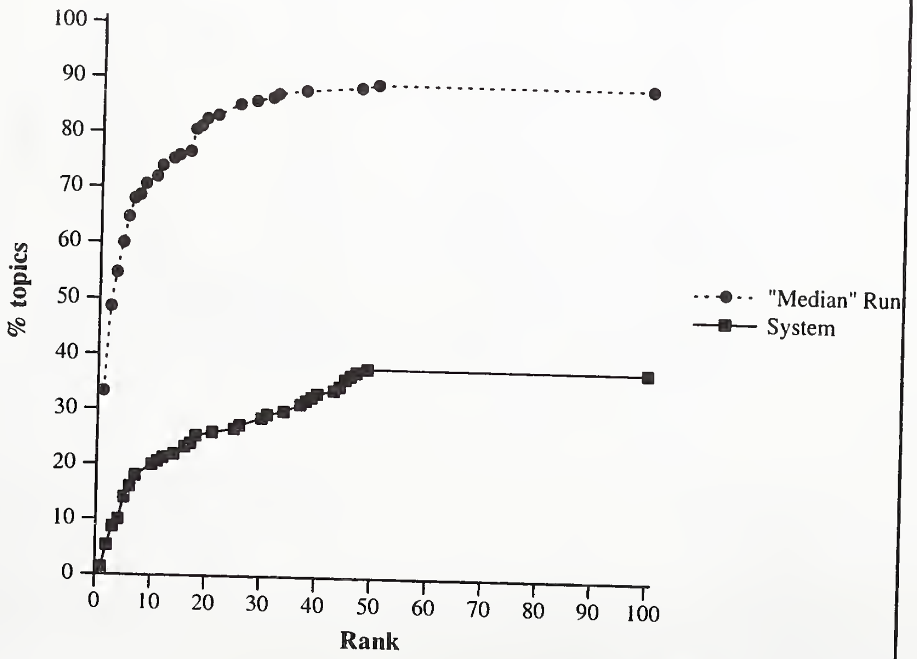
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	ajouai0202
Run Description	DOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.072
Num found at rank 1	5 (3.3%)
Num found in top 10	28 (18.7%)
Num not found in top 100	99 (66.0%)



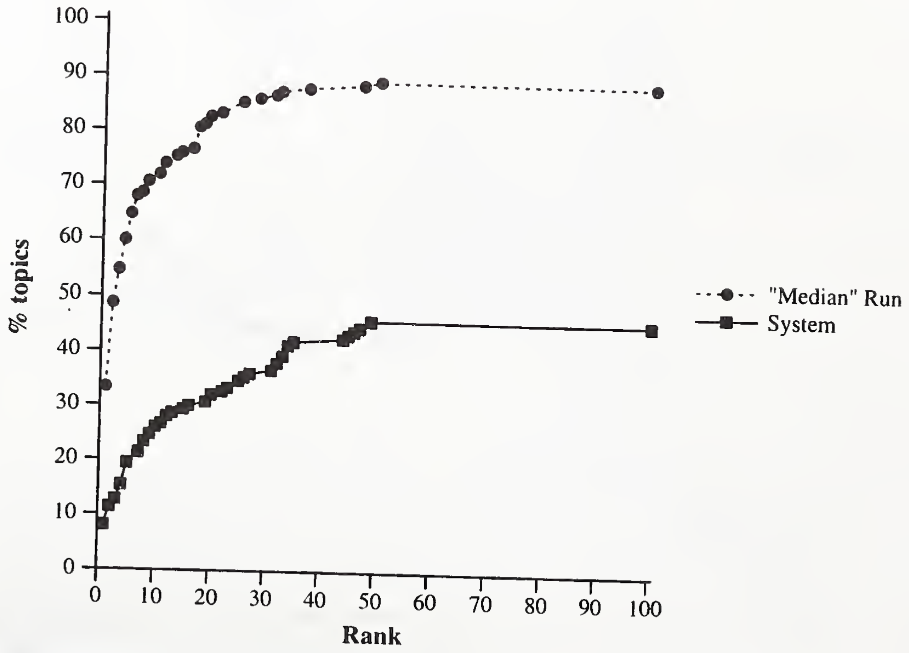
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	ajouai0203
Run Description	DOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.071
Num found at rank 1	2 (1.3%)
Num found in top 10	30 (20.0%)
Num not found in top 100	93 (62.0%)



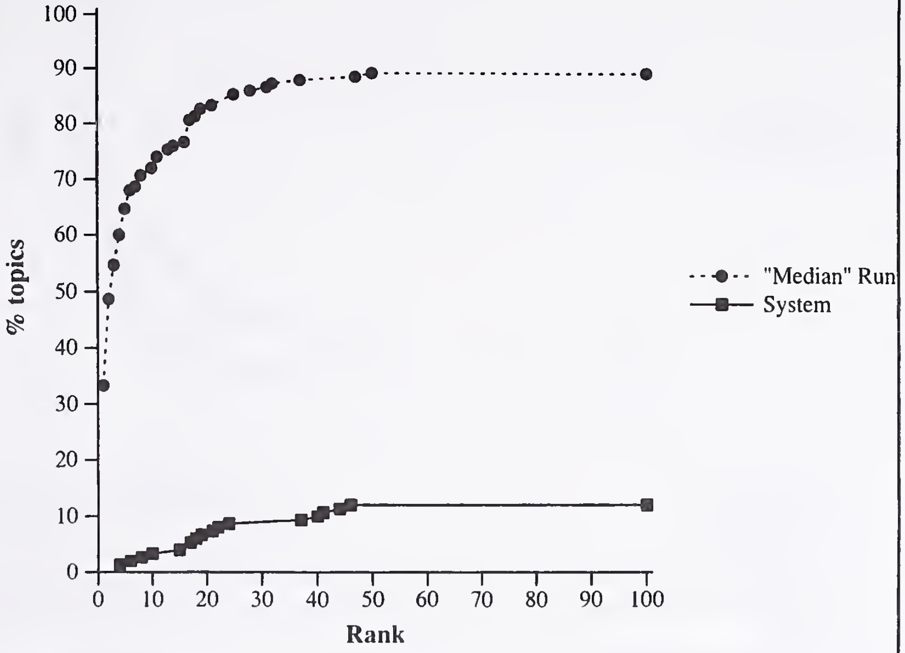
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	ajouai0204
Run Description	DOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.132
Num found at rank 1	12 (8.0%)
Num found in top 10	39 (26.0%)
Num not found in top 100	81 (54.0%)



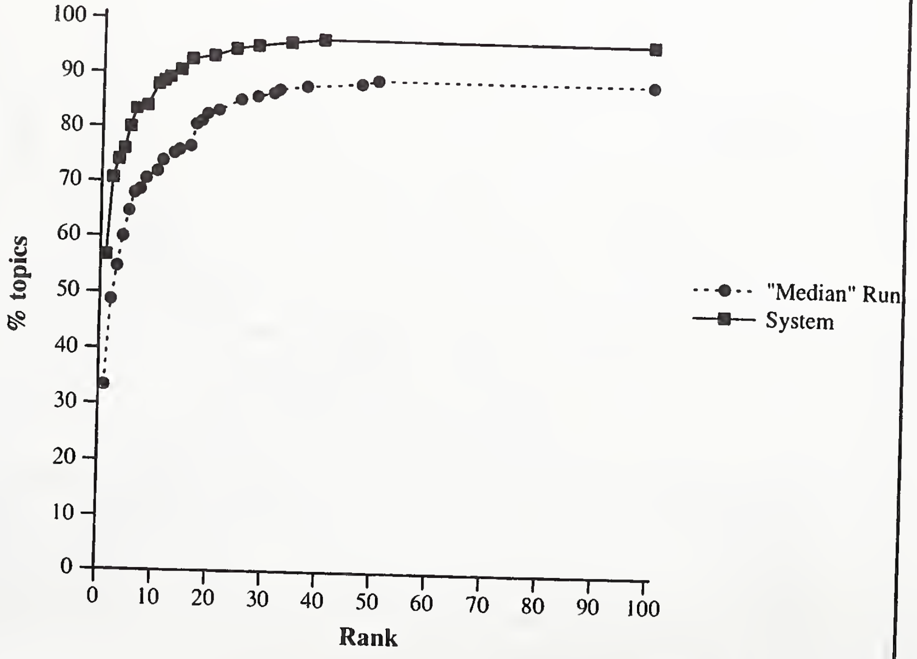
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	ajouai0205
Run Description	DOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Num topics	150
Mean reciprocal rank	0.010
Num found at rank 1	0 (0.0%)
Num found in top 10	5 (3.3%)
Num not found in top 100	132 (88.0%)



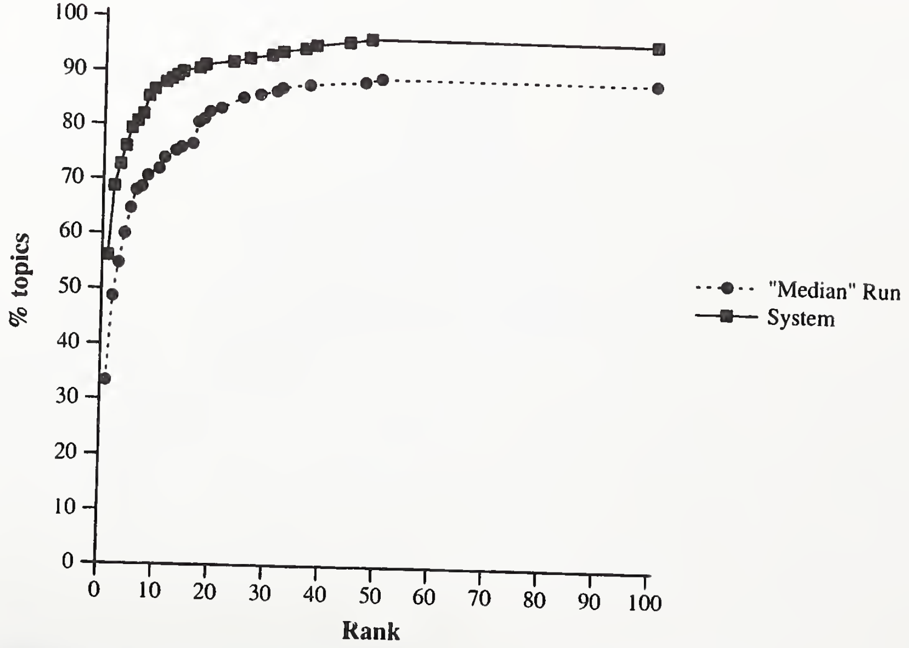
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	LmrAllEq
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.676
Num found at rank 1	85 (56.7%)
Num found in top 10	132 (88.0%)
Num not found in top 100	5 (3.3%)



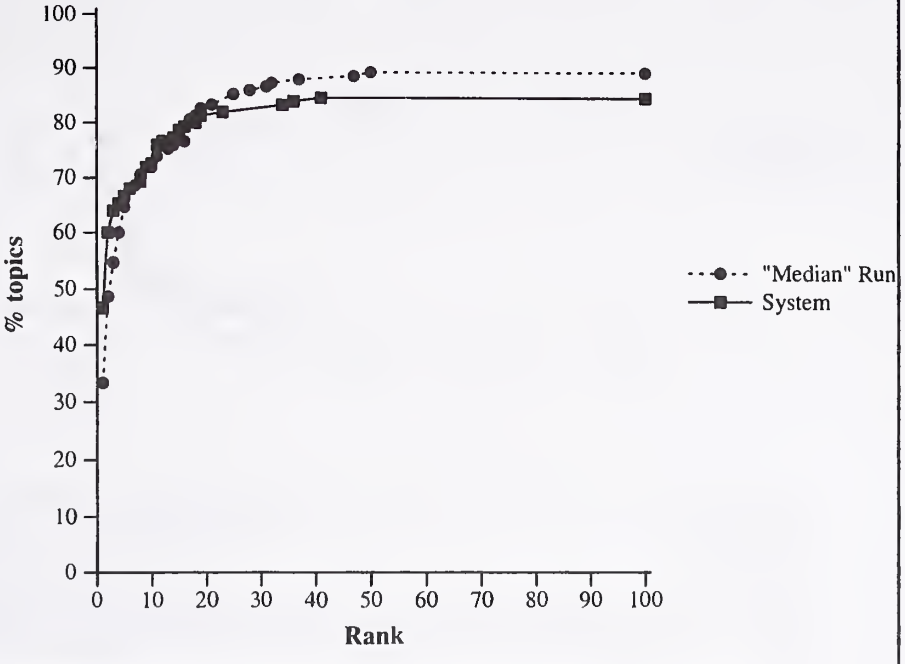
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	LmrAllEst
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.667
Num found at rank 1	84 (56.0%)
Num found in top 10	130 (86.7%)
Num not found in top 100	5 (3.3%)



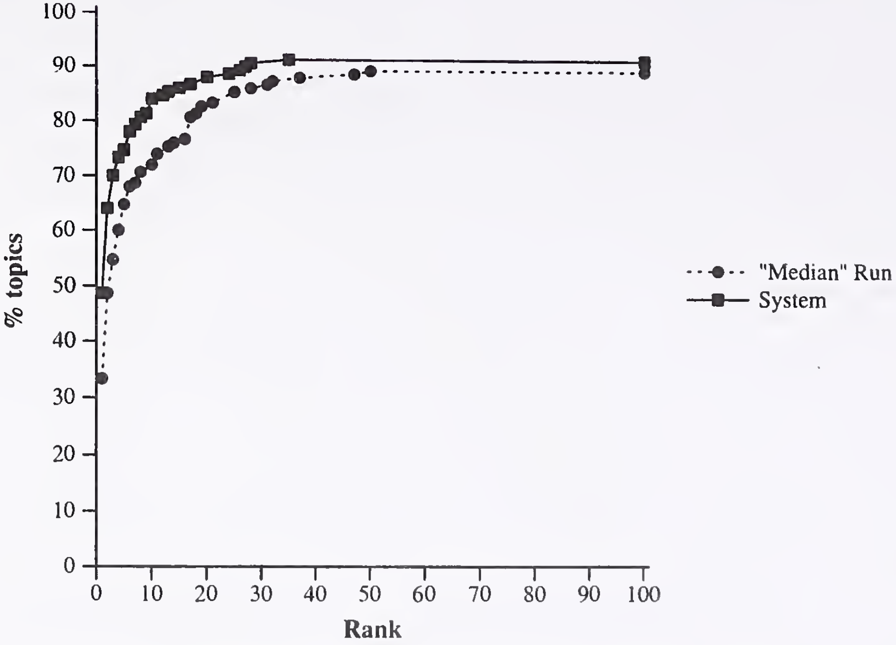
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	LmrDocStruct
Run Description	DOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.568
Num found at rank 1	70 (46.7%)
Num found in top 10	109 (72.7%)
Num not found in top 100	23 (15.3%)



Cumulative % of topics that retrieve named page by given rank

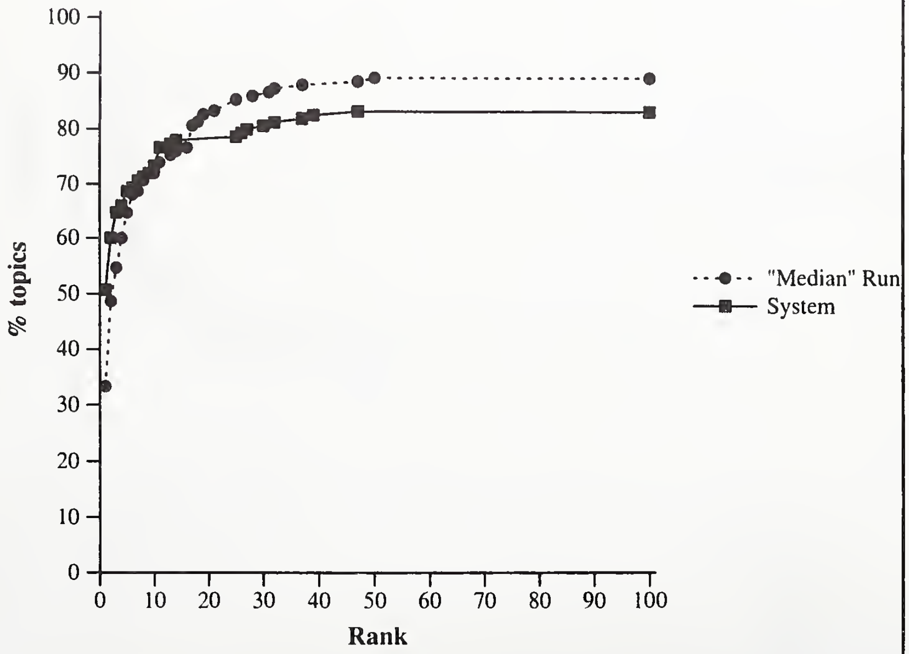
Summary Statistics	
Run ID	LmrNoStruct
Run Description	NODOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.611
Num found at rank 1	73 (48.7%)
Num found in top 10	126 (84.0%)
Num not found in top 100	13 (8.7%)



Cumulative % of topics that retrieve named page by given rank

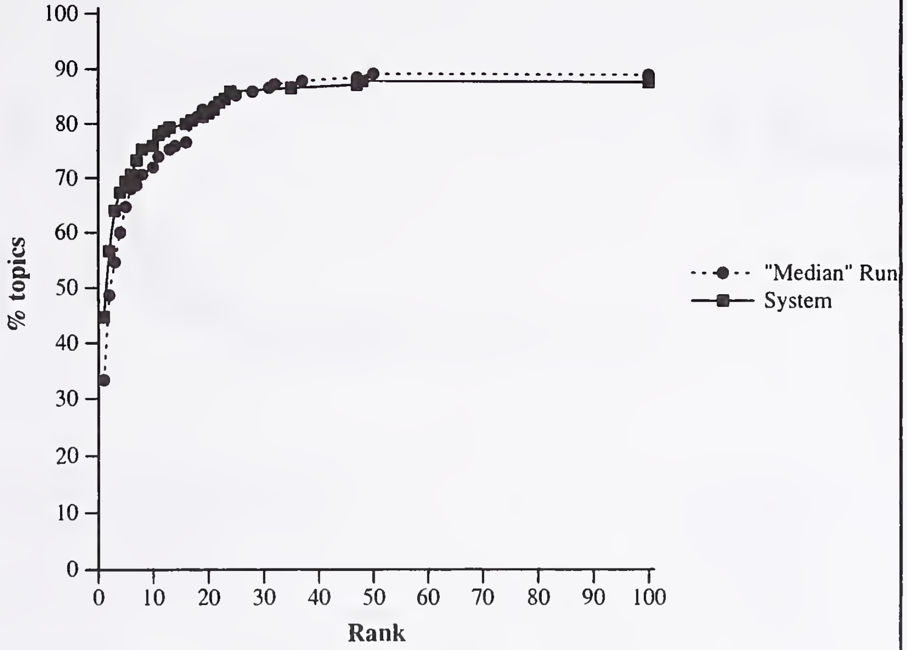
Web track, named page finding task results — Carnegie Mellon University (CMUDIR)

Summary Statistics	
Run ID	LmrSmall
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.589
Num found at rank 1	76 (50.7%)
Num found in top 10	110 (73.3%)
Num not found in top 100	25 (16.7%)



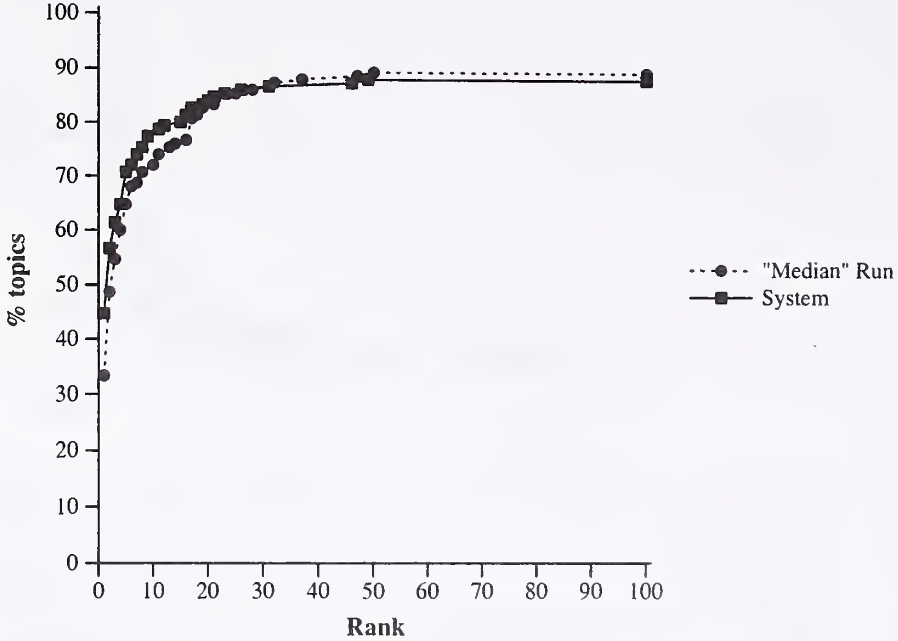
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	ictnp2
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.559
Num found at rank 1	67 (44.7%)
Num found in top 10	114 (76.0%)
Num not found in top 100	18 (12.0%)



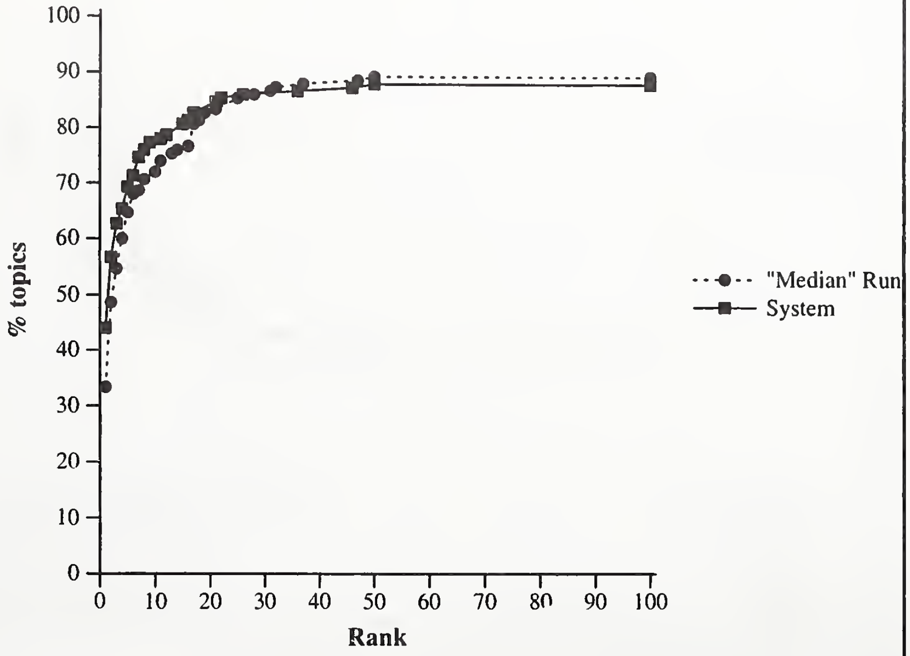
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	ictnp3
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.557
Num found at rank 1	67 (44.7%)
Num found in top 10	116 (77.3%)
Num not found in top 100	18 (12.0%)



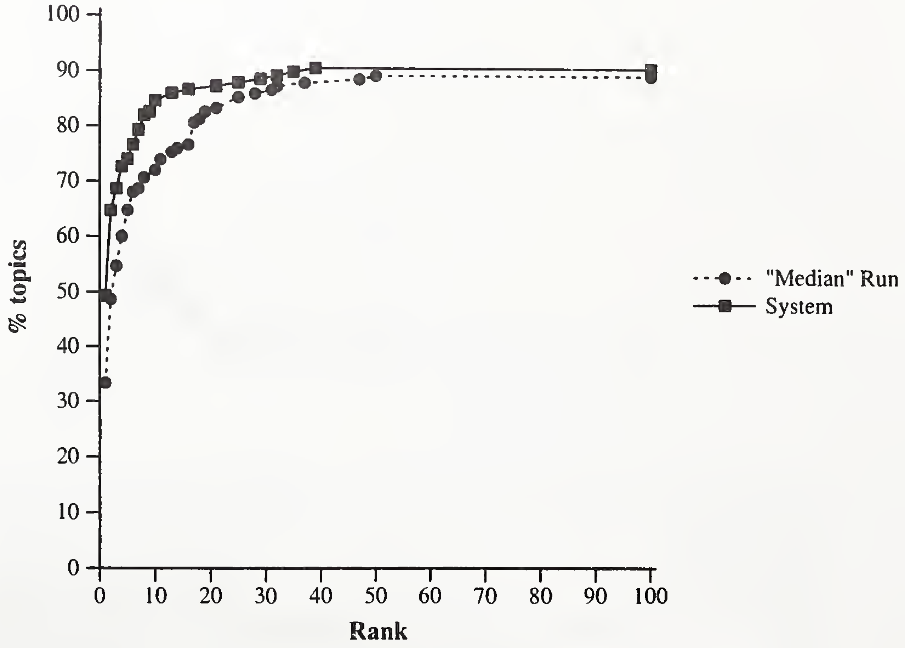
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	ictnp4
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.555
Num found at rank 1	66 (44.0%)
Num found in top 10	116 (77.3%)
Num not found in top 100	18 (12.0%)



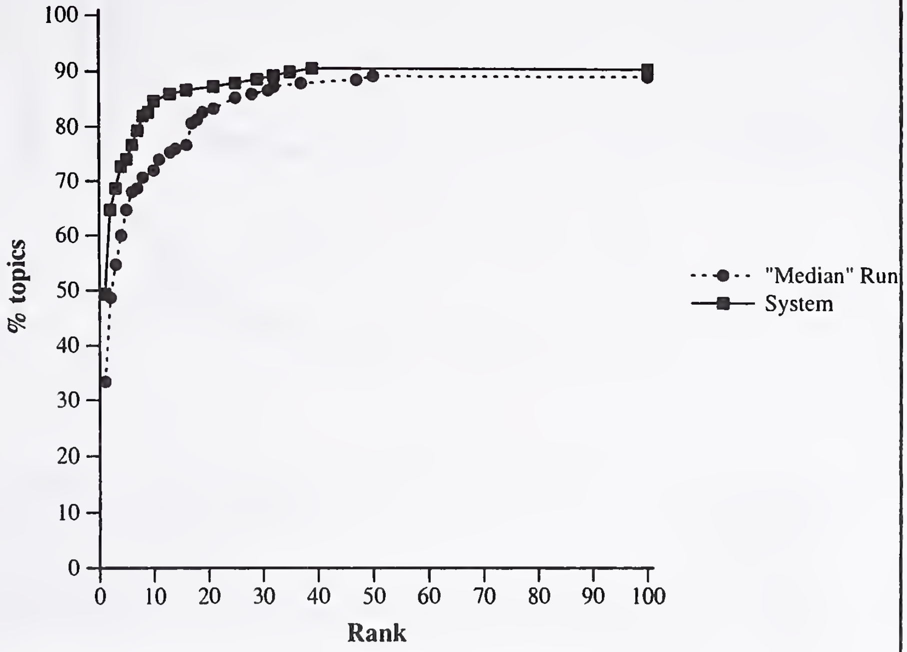
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	ictnp6
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.613
Num found at rank 1	74 (49.3%)
Num found in top 10	127 (84.7%)
Num not found in top 100	14 (9.3%)



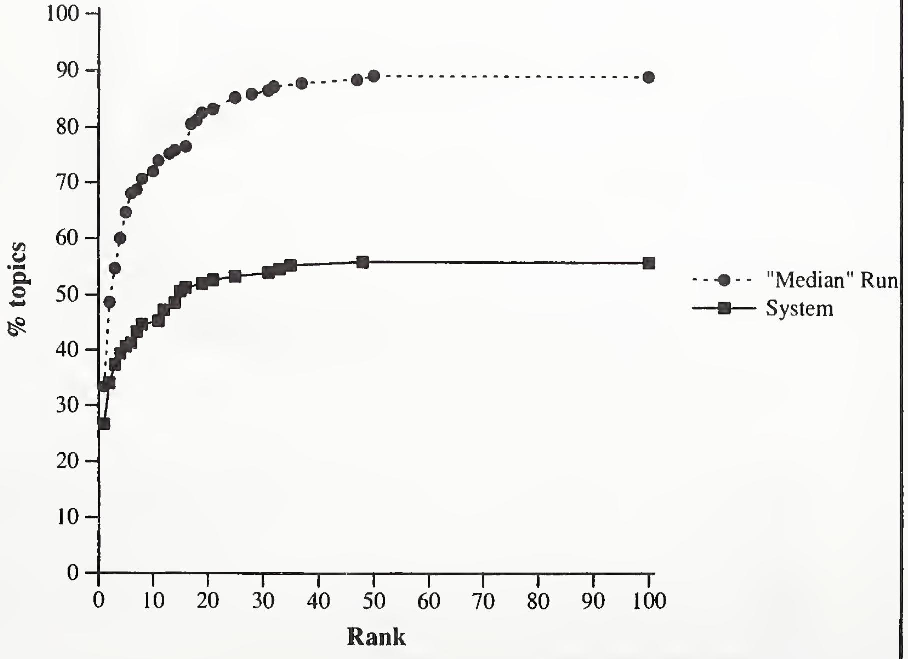
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	ictnp7
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.613
Num found at rank 1	74 (49.3%)
Num found in top 10	127 (84.7%)
Num not found in top 100	14 (9.3%)



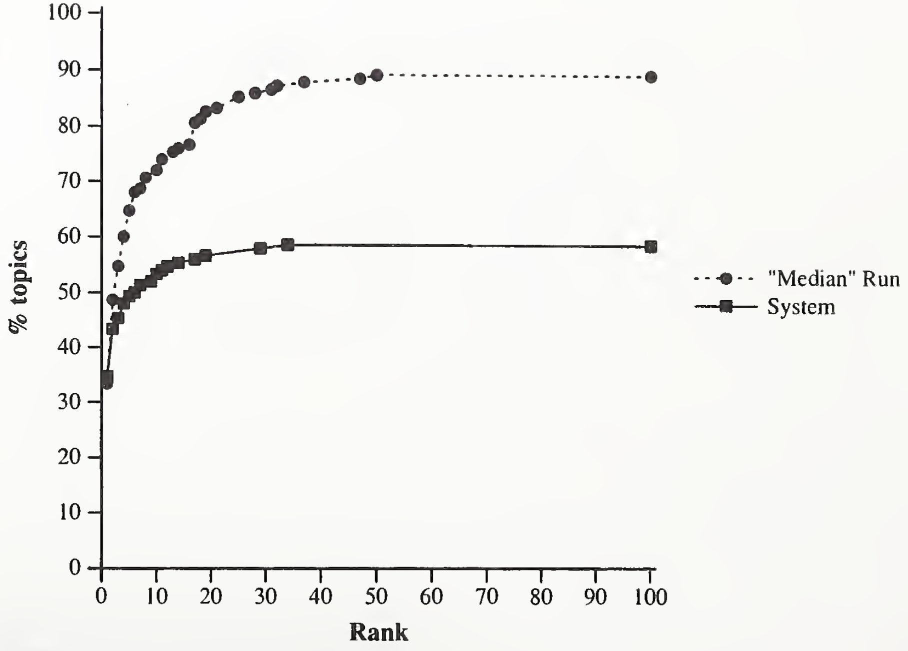
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	pltr02wt6
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.334
Num found at rank 1	40 (26.7%)
Num found in top 10	67 (44.7%)
Num not found in top 100	66 (44.0%)



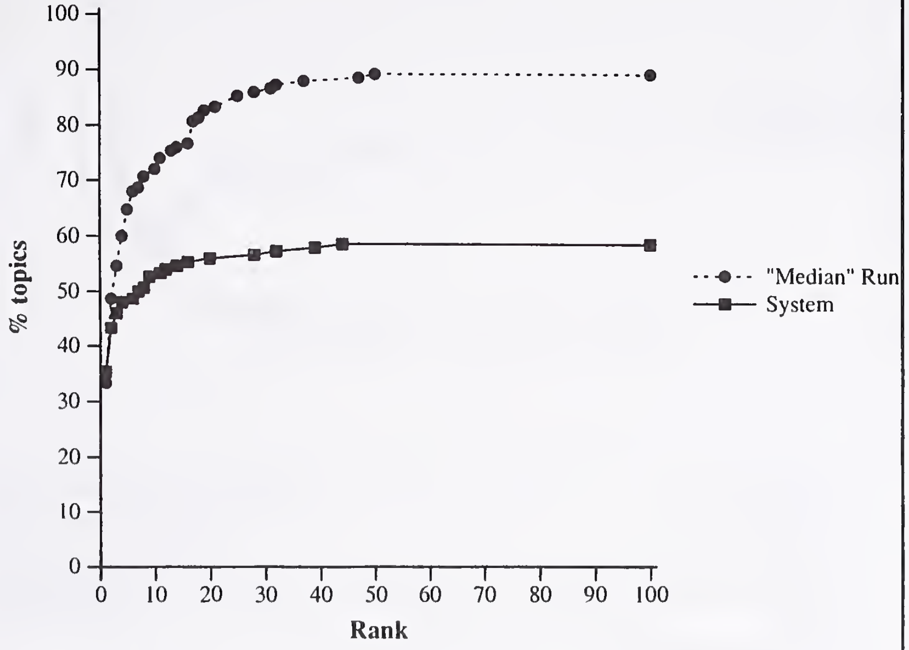
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	pltr02wt7
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.414
Num found at rank 1	52 (34.7%)
Num found in top 10	80 (53.3%)
Num not found in top 100	62 (41.3%)



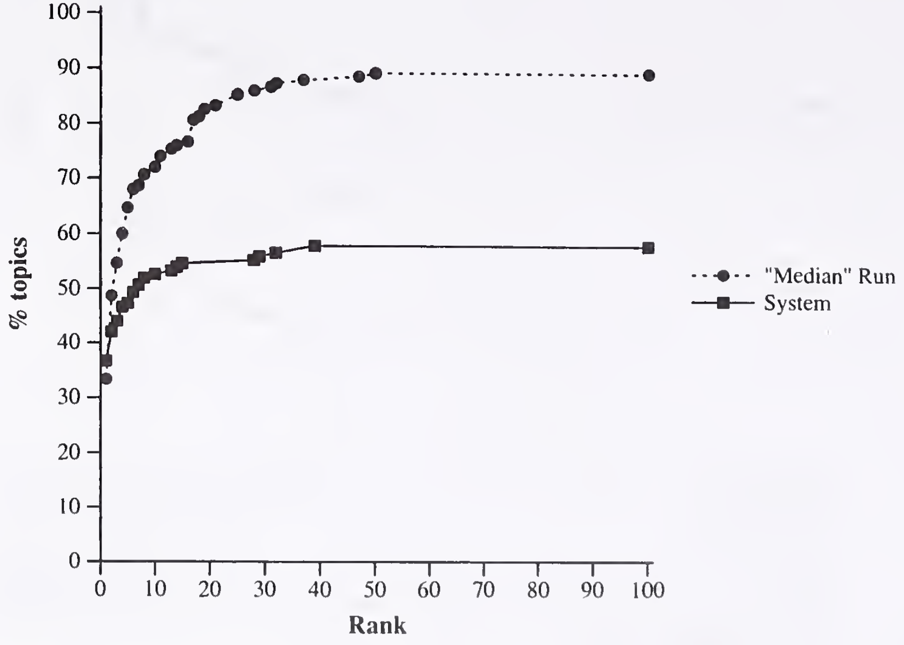
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	pltr02wt8
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.416
Num found at rank 1	53 (35.3%)
Num found in top 10	79 (52.7%)
Num not found in top 100	62 (41.3%)



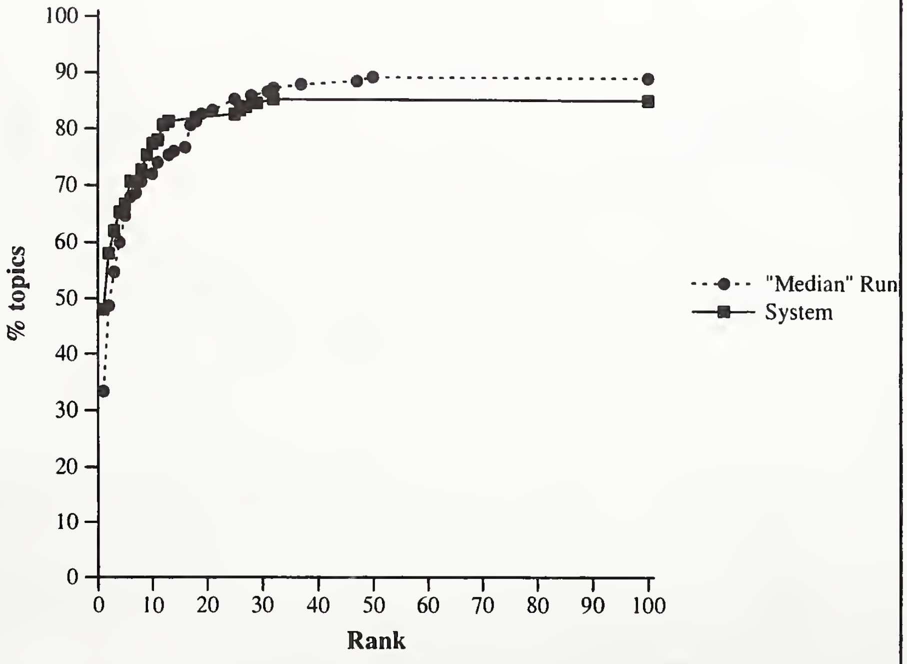
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	pltr02wt9
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.418
Num found at rank 1	55 (36.7%)
Num found in top 10	79 (52.7%)
Num not found in top 100	63 (42.0%)



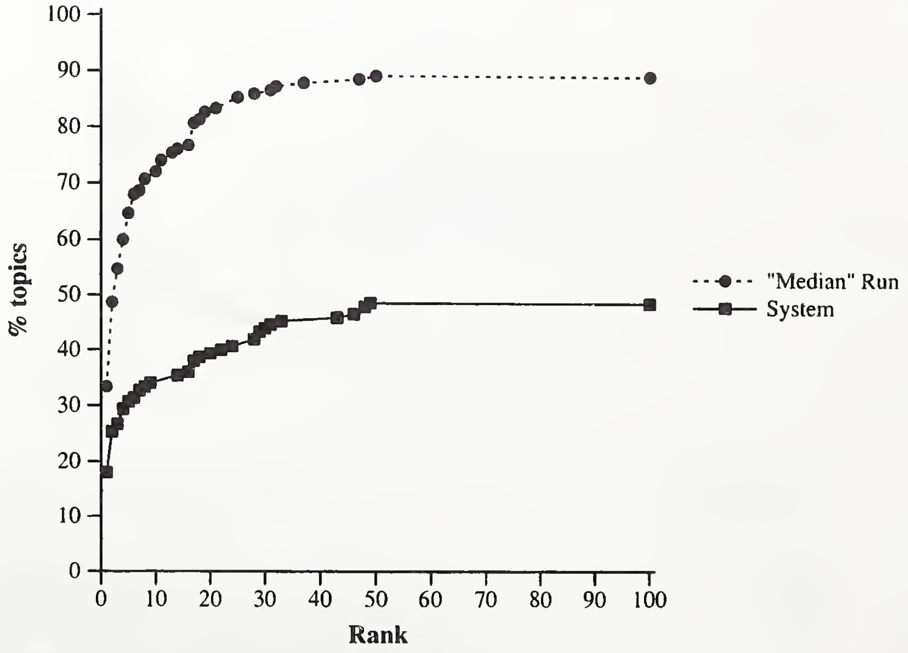
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	csiro02np01
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.573
Num found at rank 1	72 (48.0%)
Num found in top 10	116 (77.3%)
Num not found in top 100	22 (14.7%)



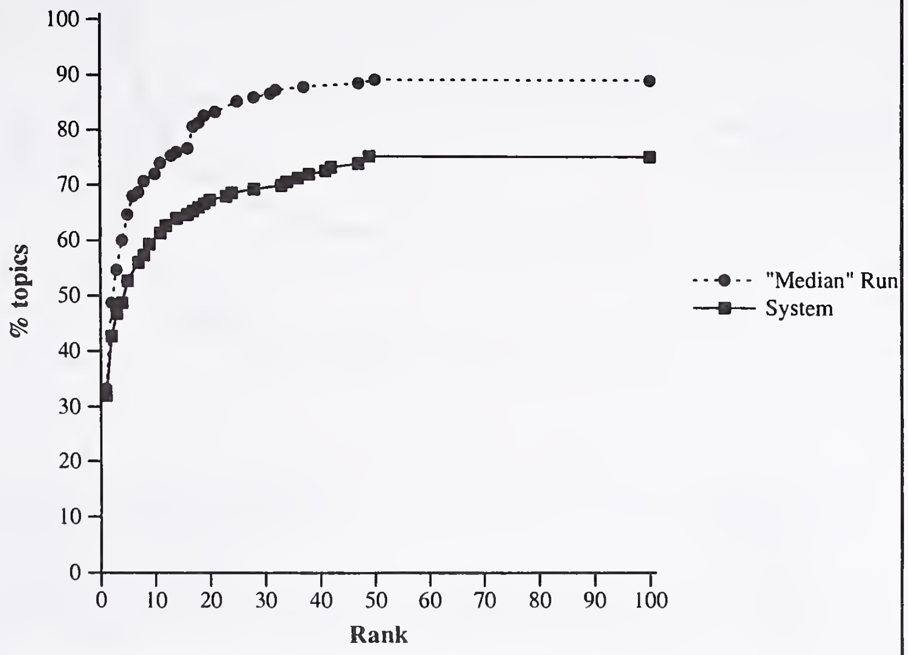
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	csiro02np02
Run Description	NODOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.241
Num found at rank 1	27 (18.0%)
Num found in top 10	51 (34.0%)
Num not found in top 100	77 (51.3%)



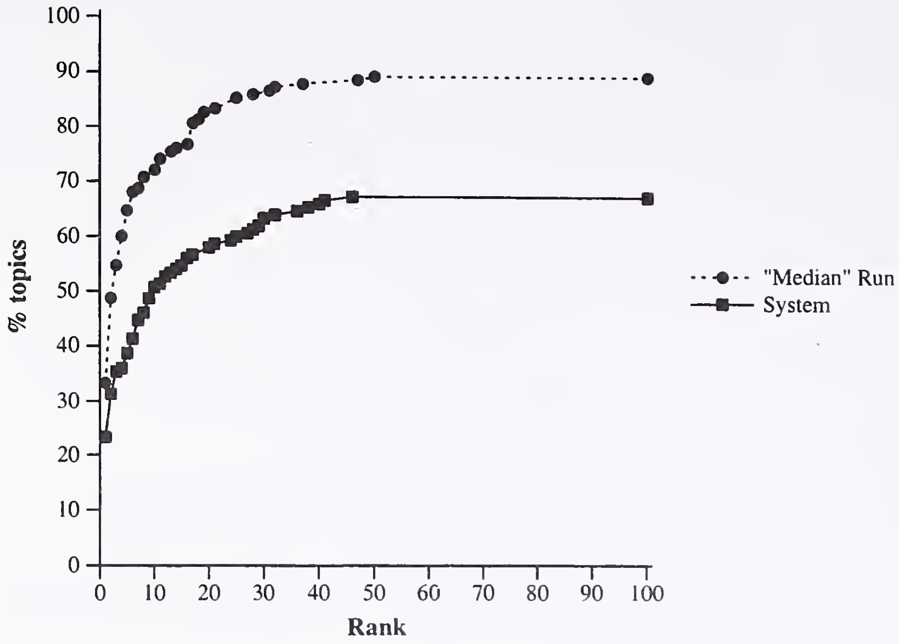
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	csiro02np03
Run Description	DOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.416
Num found at rank 1	48 (32.0%)
Num found in top 10	89 (59.3%)
Num not found in top 100	37 (24.7%)



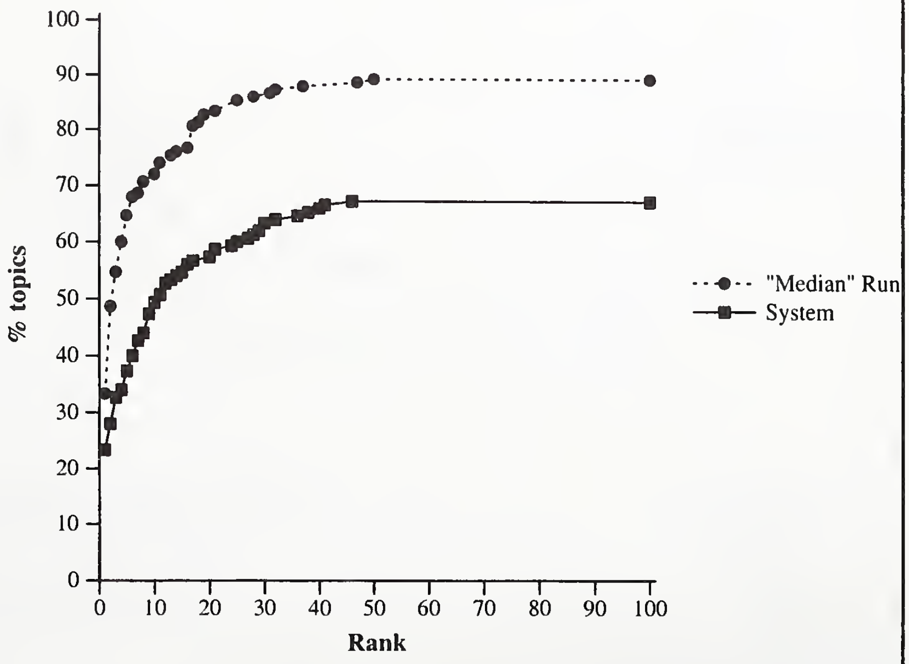
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	csiro02np04
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.318
Num found at rank 1	35 (23.3%)
Num found in top 10	76 (50.7%)
Num not found in top 100	49 (32.7%)



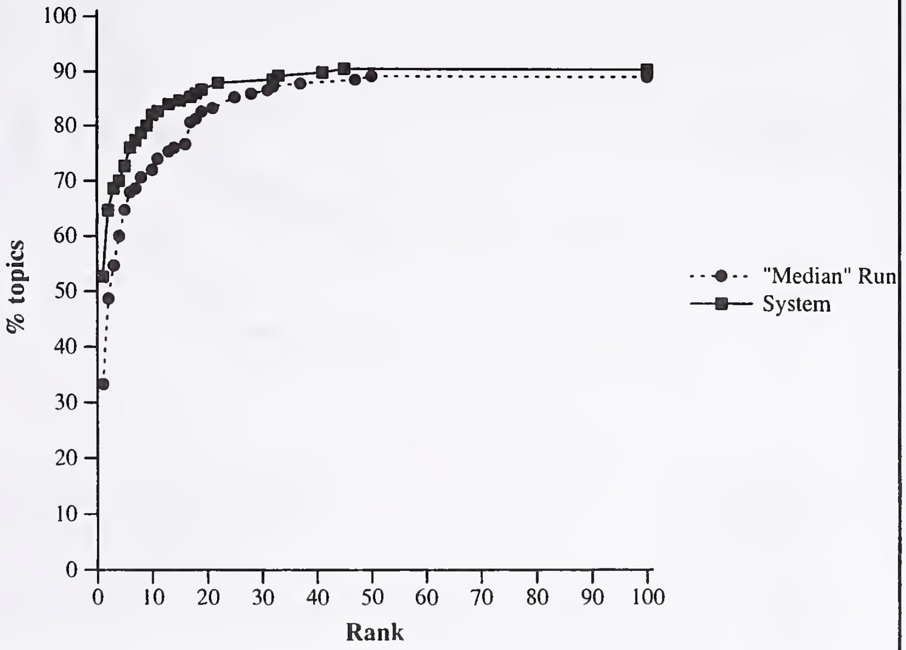
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	csiro02np16
Run Description	DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Num topics	150
Mean reciprocal rank	0.307
Num found at rank 1	35 (23.3%)
Num found in top 10	74 (49.3%)
Num not found in top 100	49 (32.7%)



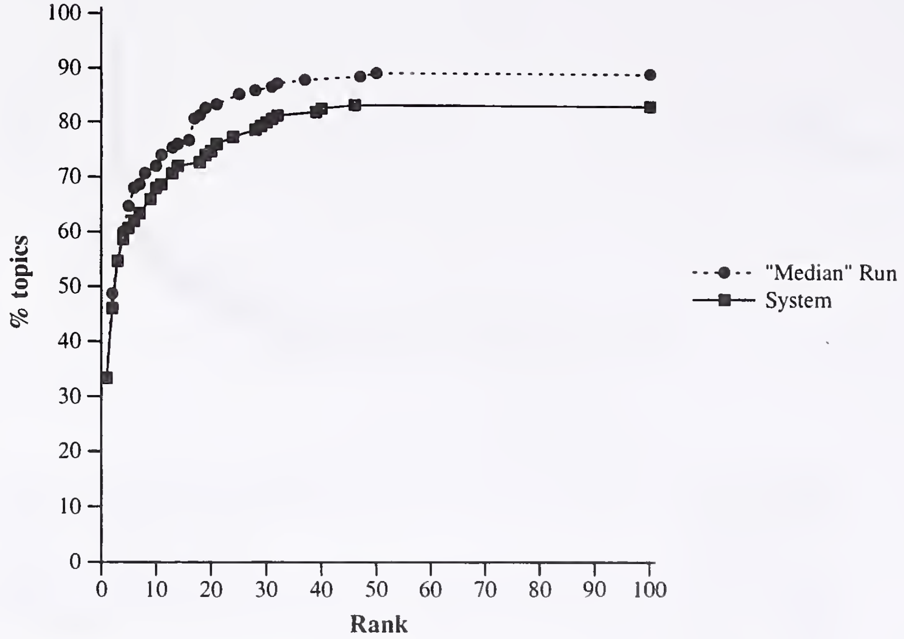
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	hum02pd
Run Description	DOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.626
Num found at rank 1	79 (52.7%)
Num found in top 10	123 (82.0%)
Num not found in top 100	14 (9.3%)



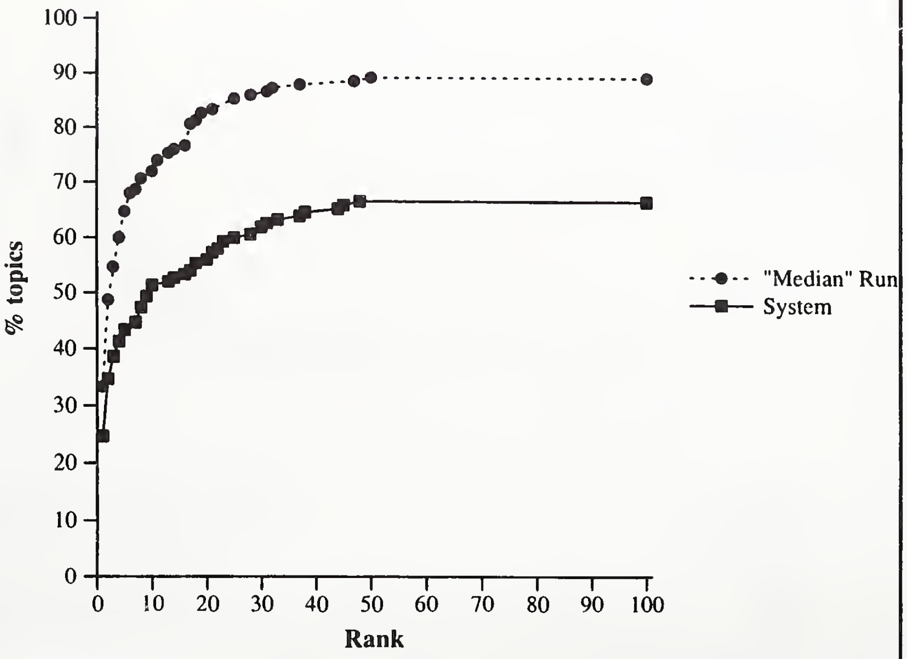
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	hum02ud
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.456
Num found at rank 1	50 (33.3%)
Num found in top 10	102 (68.0%)
Num not found in top 100	25 (16.7%)



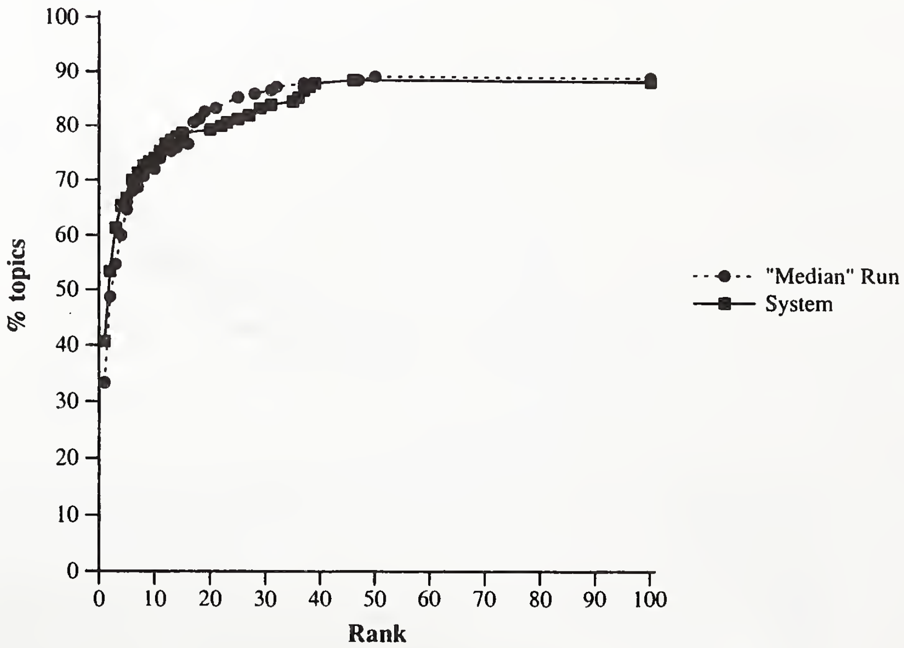
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	hum02uhp
Run Description	DOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.337
Num found at rank 1	37 (24.7%)
Num found in top 10	77 (51.3%)
Num not found in top 100	50 (33.3%)



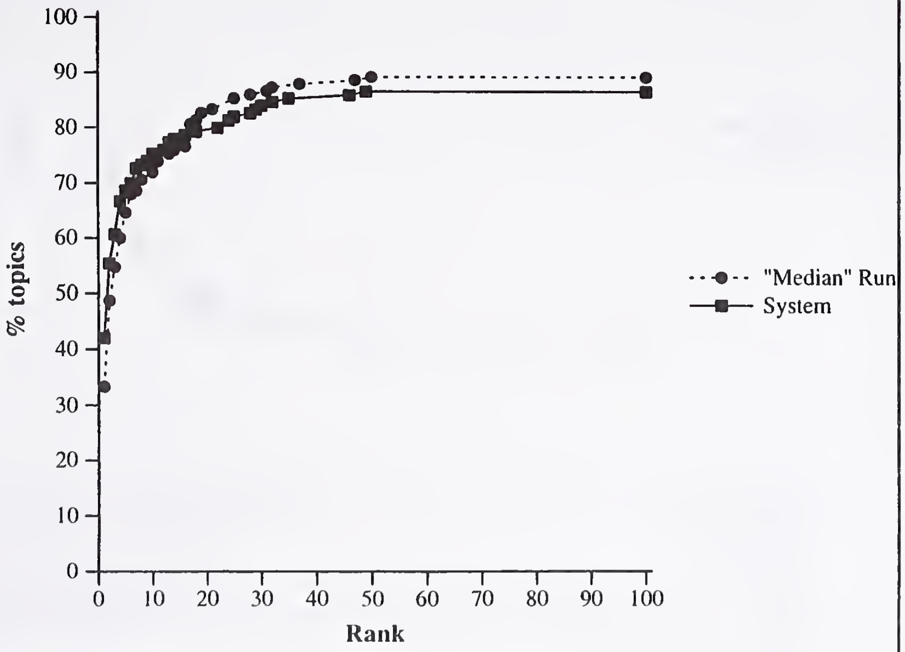
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	hum02up
Run Description	DOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.527
Num found at rank 1	61 (40.7%)
Num found in top 10	111 (74.0%)
Num not found in top 100	17 (11.3%)



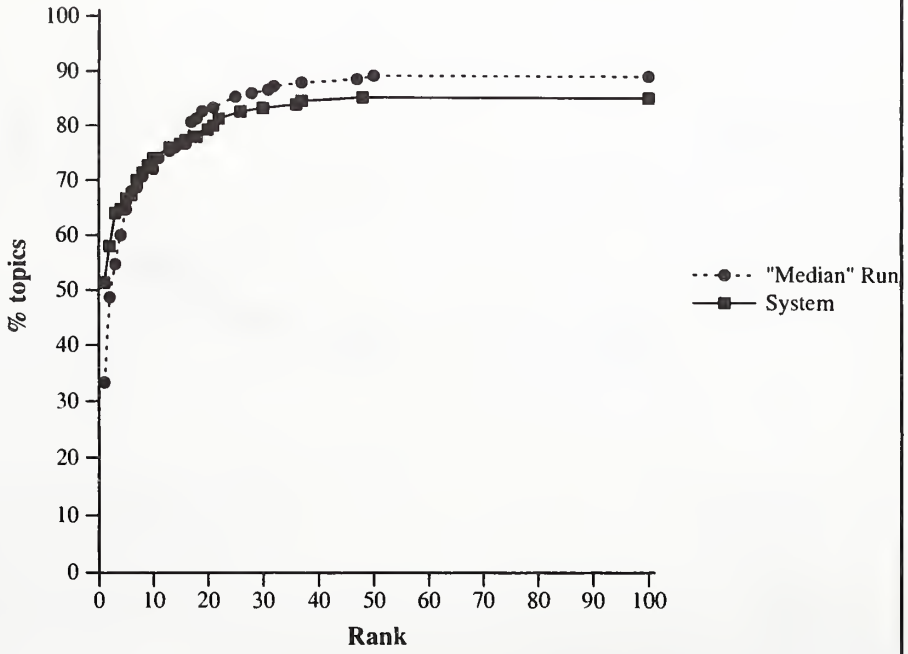
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	hum02upd
Run Description	DOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.538
Num found at rank 1	63 (42.0%)
Num found in top 10	113 (75.3%)
Num not found in top 100	20 (13.3%)



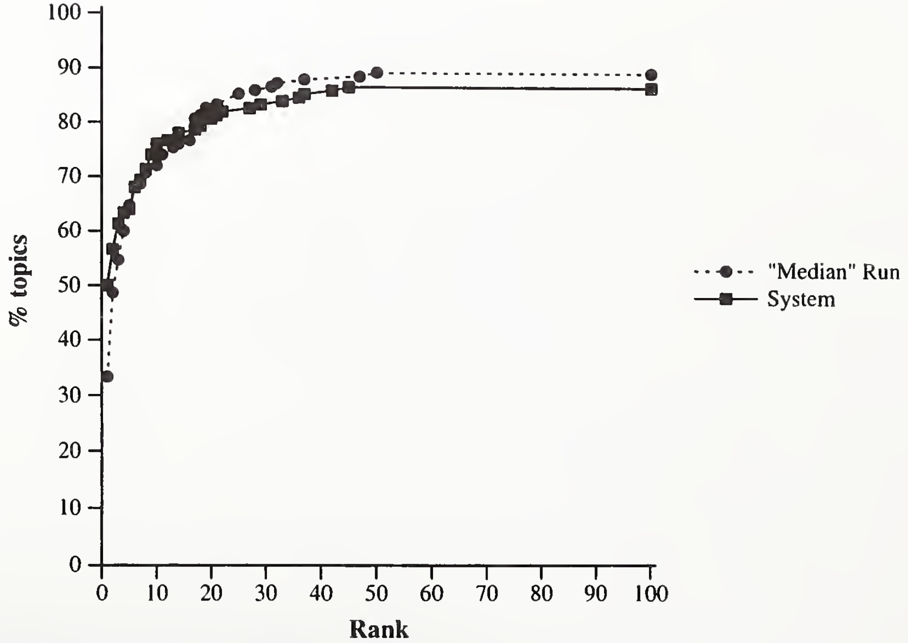
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	iit02b
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.587
Num found at rank 1	77 (51.3%)
Num found in top 10	111 (74.0%)
Num not found in top 100	22 (14.7%)



Cumulative % of topics that retrieve named page by given rank

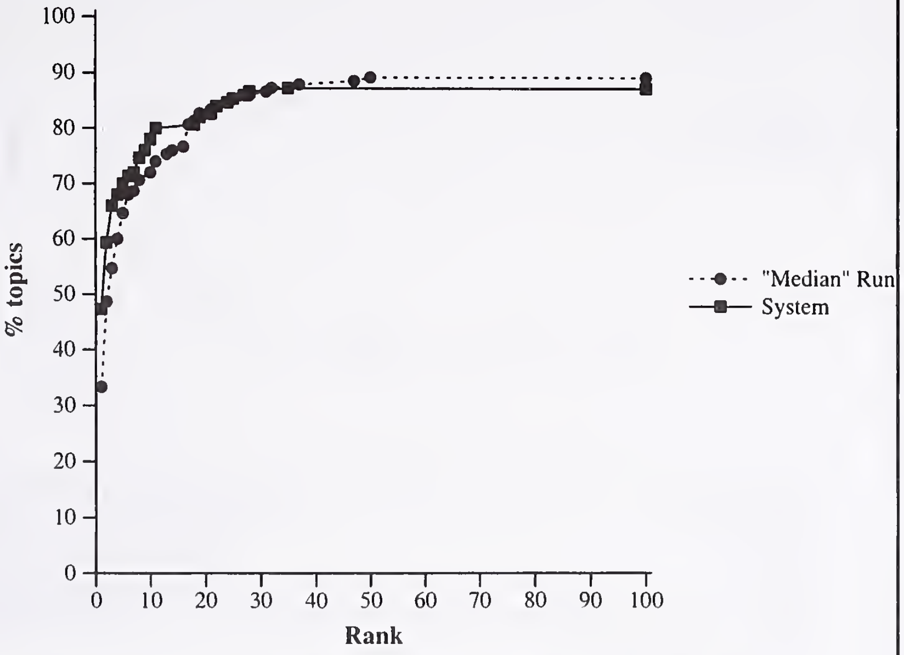
Summary Statistics	
Run ID	iit02tf
Run Description	DOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.576
Num found at rank 1	75 (50.0%)
Num found in top 10	114 (76.0%)
Num not found in top 100	20 (13.3%)



Cumulative % of topics that retrieve named page by given rank

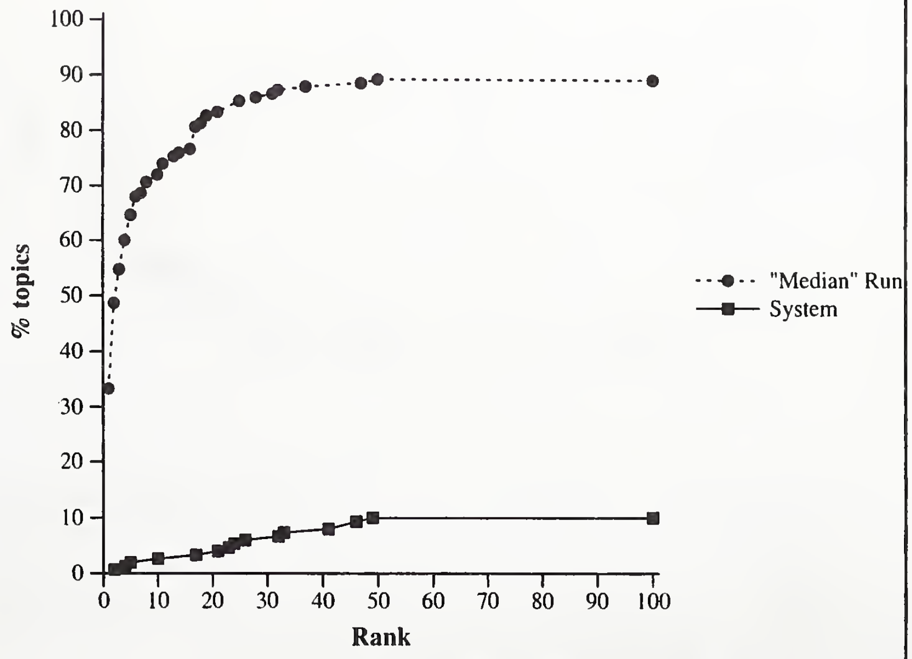
Web track, named page finding task results — IIT Information Retrieval Lab

Summary Statistics	
Run ID	iit02tfa
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.580
Num found at rank 1	71 (47.3%)
Num found in top 10	117 (78.0%)
Num not found in top 100	19 (12.7%)



Cumulative % of topics that retrieve named page by given rank

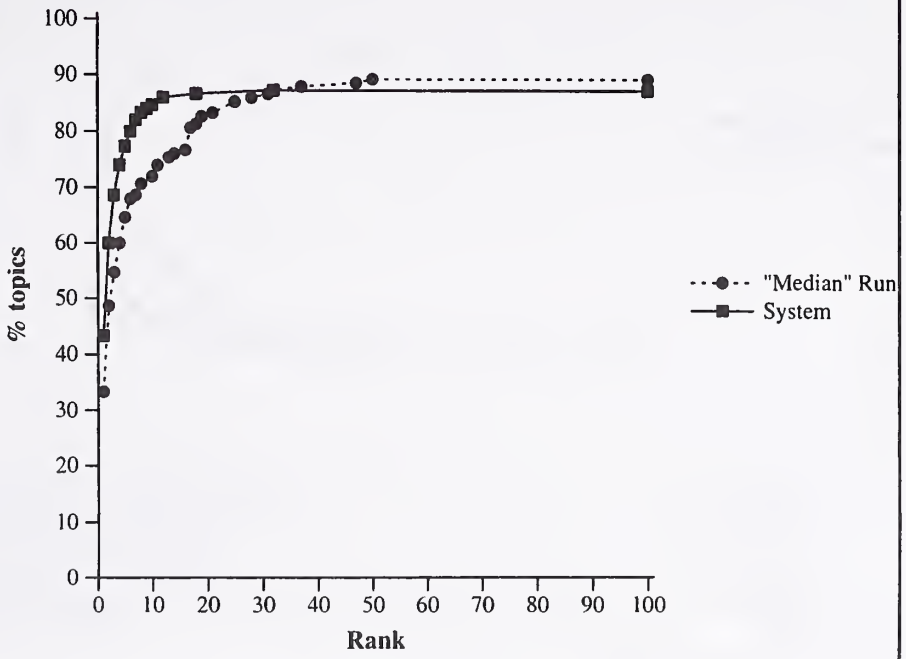
Summary Statistics	
Run ID	KUHPF0201
Run Description	NODOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.010
Num found at rank 1	0 (0.0%)
Num found in top 10	4 (2.7%)
Num not found in top 100	135 (90.0%)



Cumulative % of topics that retrieve named page by given rank

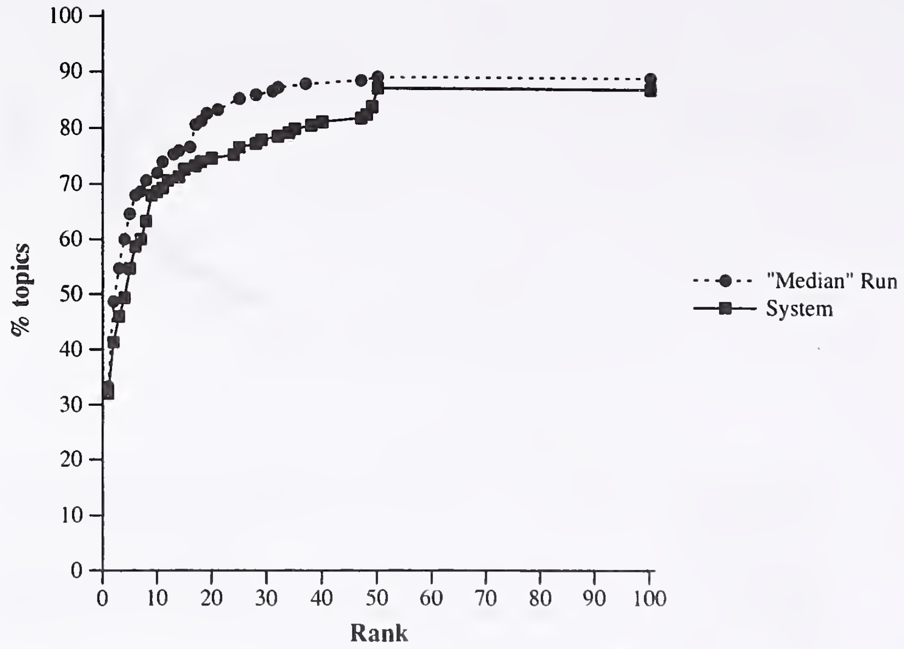
Web track, named page finding task results — Laboratory for Information Technology (KRDL)

Summary Statistics	
Run ID	litlink
Run Description	DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Num topics	150
Mean reciprocal rank	0.578
Num found at rank 1	65 (43.3%)
Num found in top 10	127 (84.7%)
Num not found in top 100	19 (12.7%)



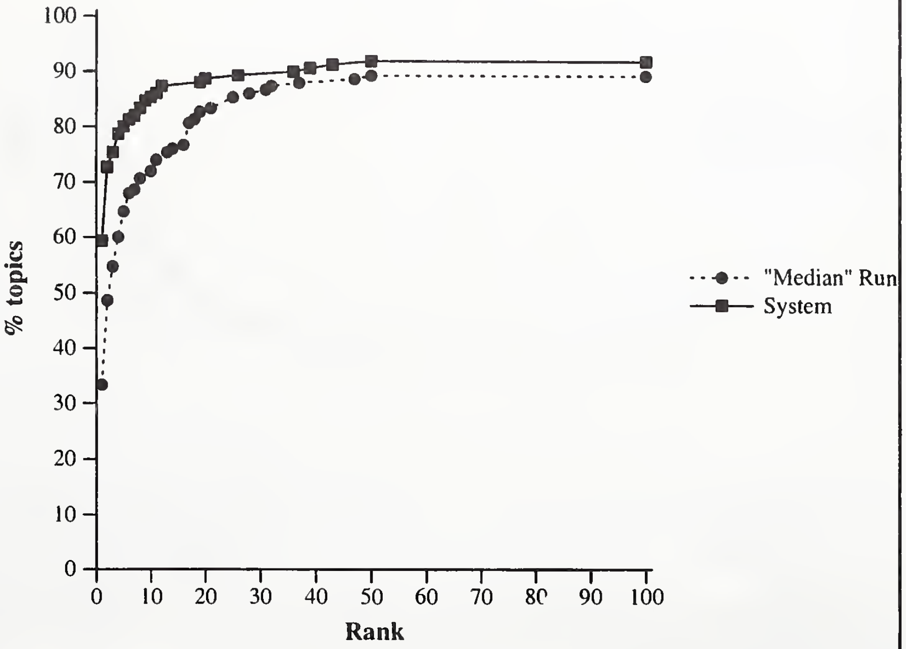
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	littext
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.427
Num found at rank 1	48 (32.0%)
Num found in top 10	103 (68.7%)
Num not found in top 100	19 (12.7%)



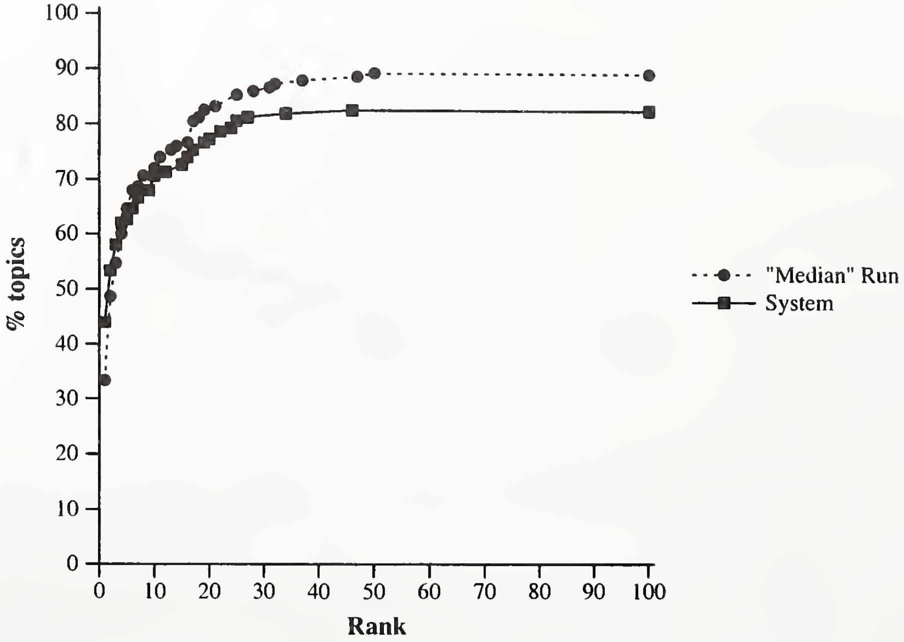
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	thunp1
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.690
Num found at rank 1	89 (59.3%)
Num found in top 10	128 (85.3%)
Num not found in top 100	12 (8.0%)



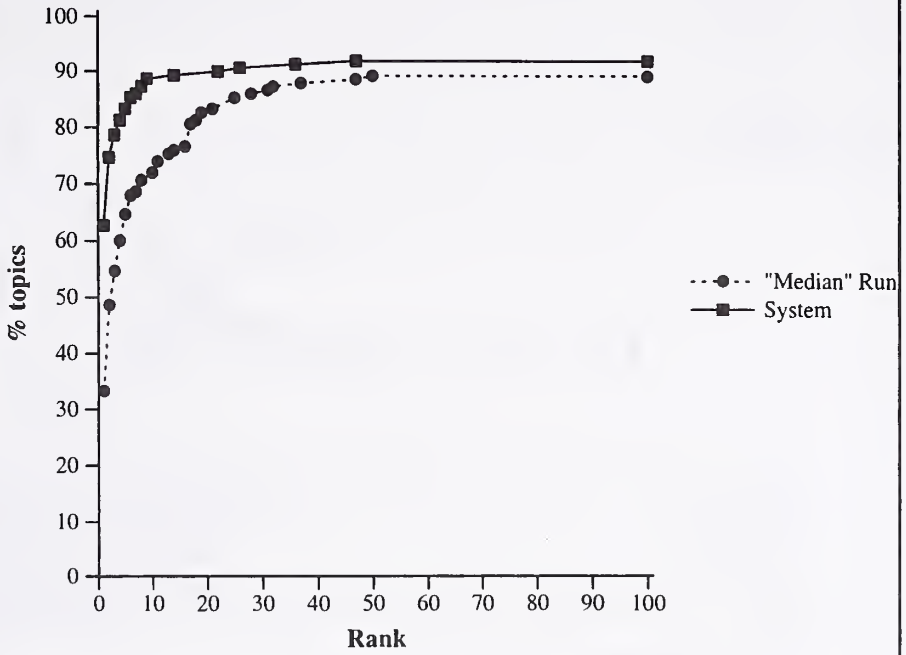
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	thunp2
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.530
Num found at rank 1	66 (44.0%)
Num found in top 10	106 (70.7%)
Num not found in top 100	26 (17.3%)



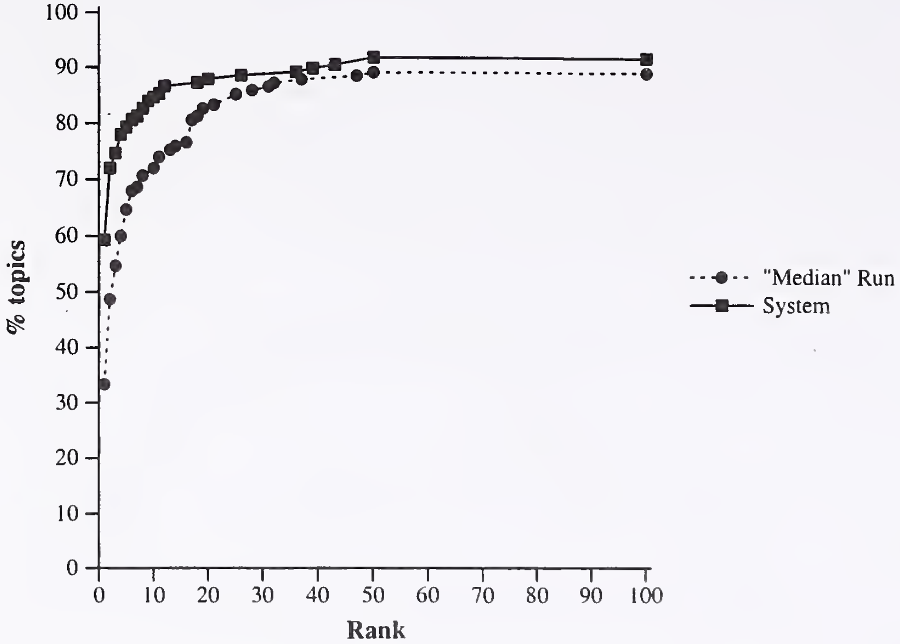
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	thunp3
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.719
Num found at rank 1	94 (62.7%)
Num found in top 10	133 (88.7%)
Num not found in top 100	12 (8.0%)



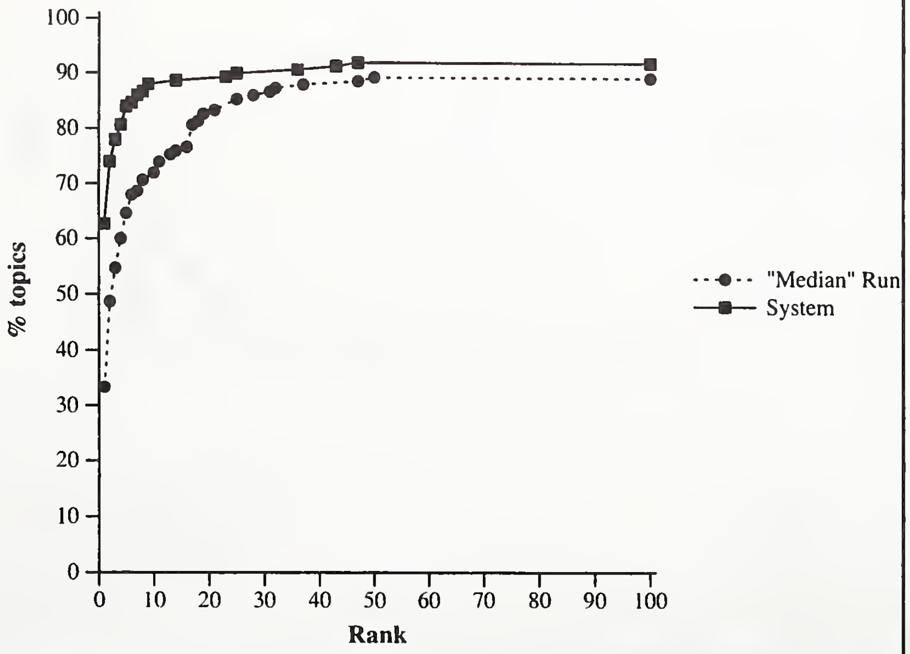
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	thunp4
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.687
Num found at rank 1	89 (59.3%)
Num found in top 10	127 (84.7%)
Num not found in top 100	12 (8.0%)



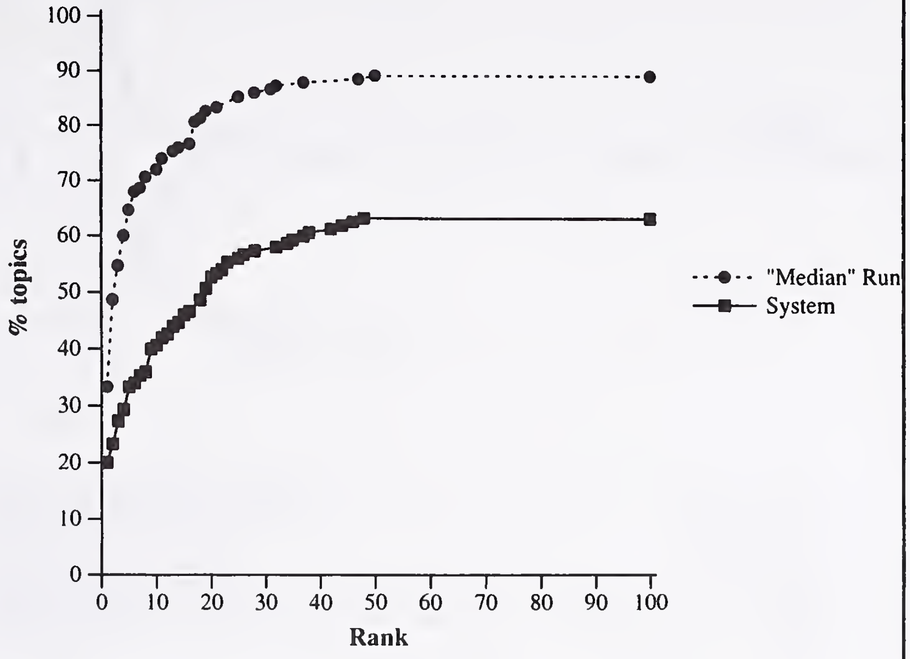
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	thunp5
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.717
Num found at rank 1	94 (62.7%)
Num found in top 10	132 (88.0%)
Num not found in top 100	12 (8.0%)



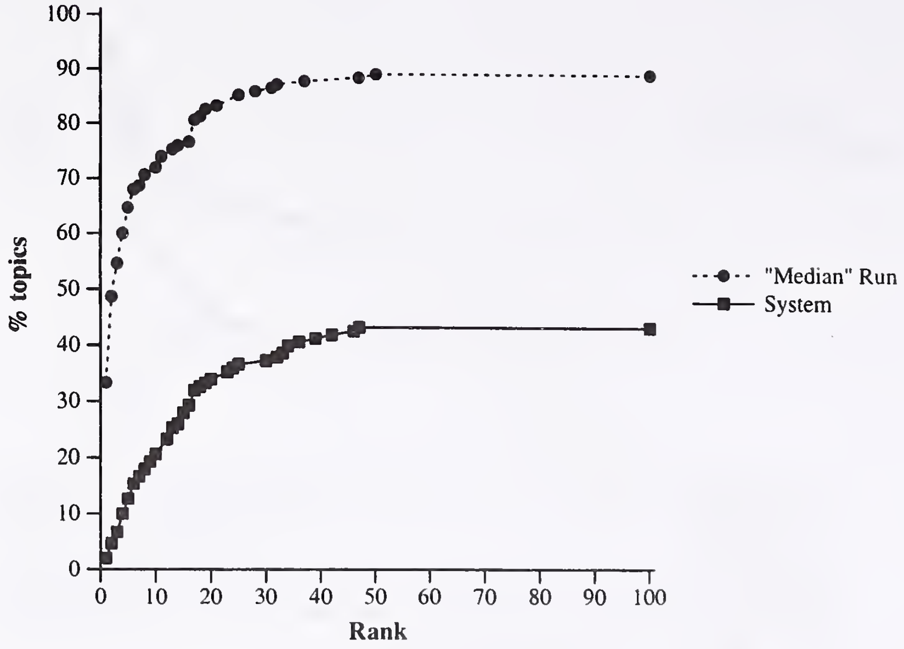
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	pirc2Wnp1
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.263
Num found at rank 1	30 (20.0%)
Num found in top 10	61 (40.7%)
Num not found in top 100	55 (36.7%)



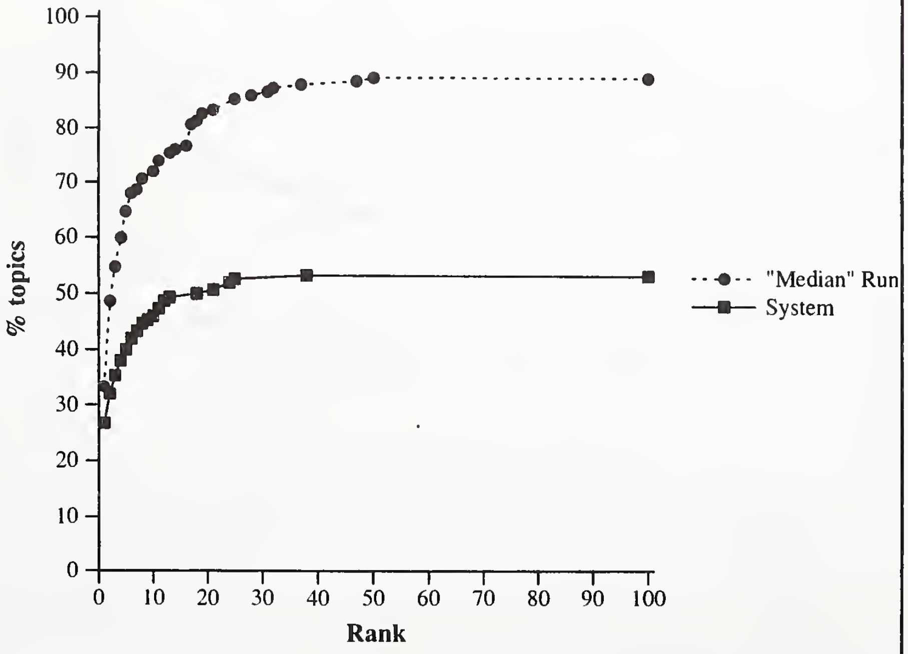
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	pirc2Wnp2
Run Description	DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Num topics	150
Mean reciprocal rank	0.076
Num found at rank 1	3 (2.0%)
Num found in top 10	31 (20.7%)
Num not found in top 100	85 (56.7%)



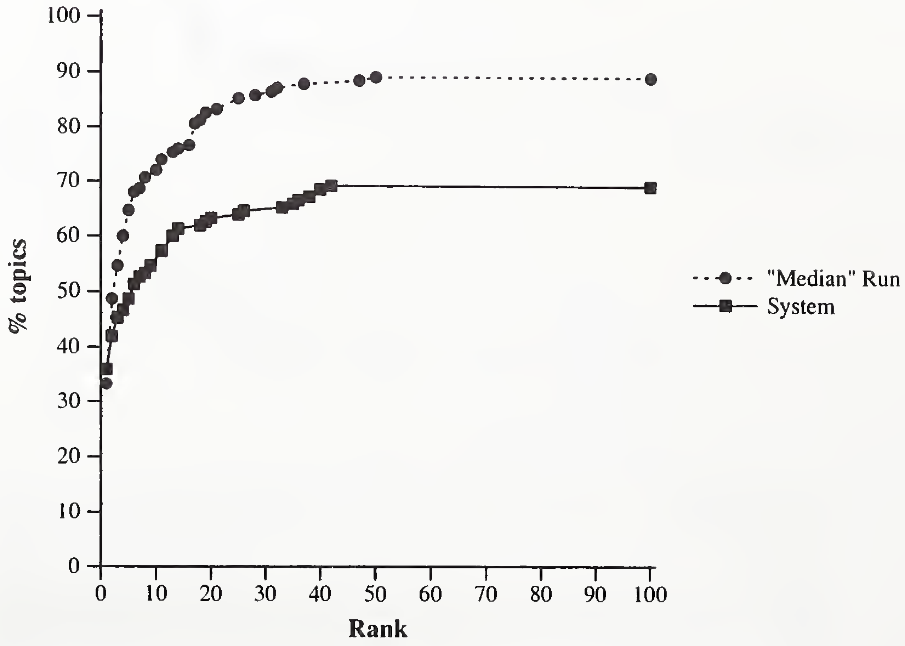
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	UAmsT02WnA
Run Description	NODOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.328
Num found at rank 1	40 (26.7%)
Num found in top 10	69 (46.0%)
Num not found in top 100	70 (46.7%)



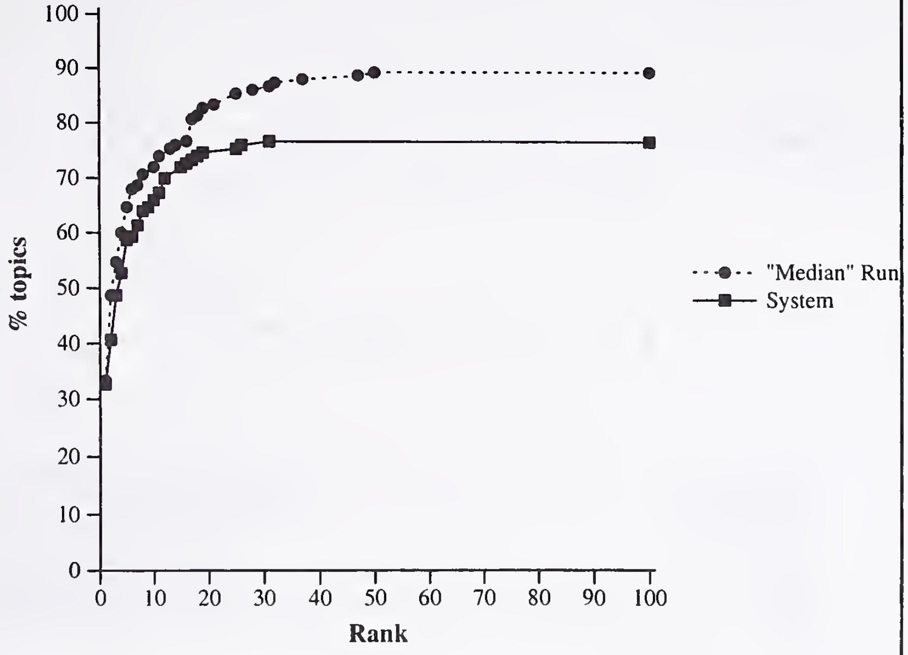
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	UAmsT02WnTl
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.425
Num found at rank 1	54 (36.0%)
Num found in top 10	82 (54.7%)
Num not found in top 100	46 (30.7%)



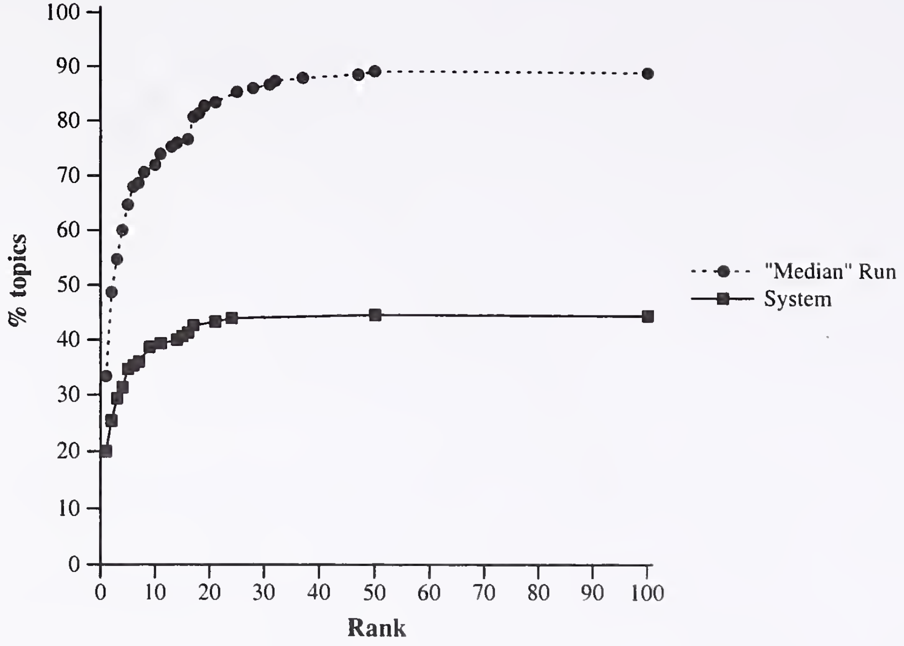
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	UAmsT02WnTlA
Run Description	NODOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.432
Num found at rank 1	49 (32.7%)
Num found in top 10	99 (66.0%)
Num not found in top 100	35 (23.3%)



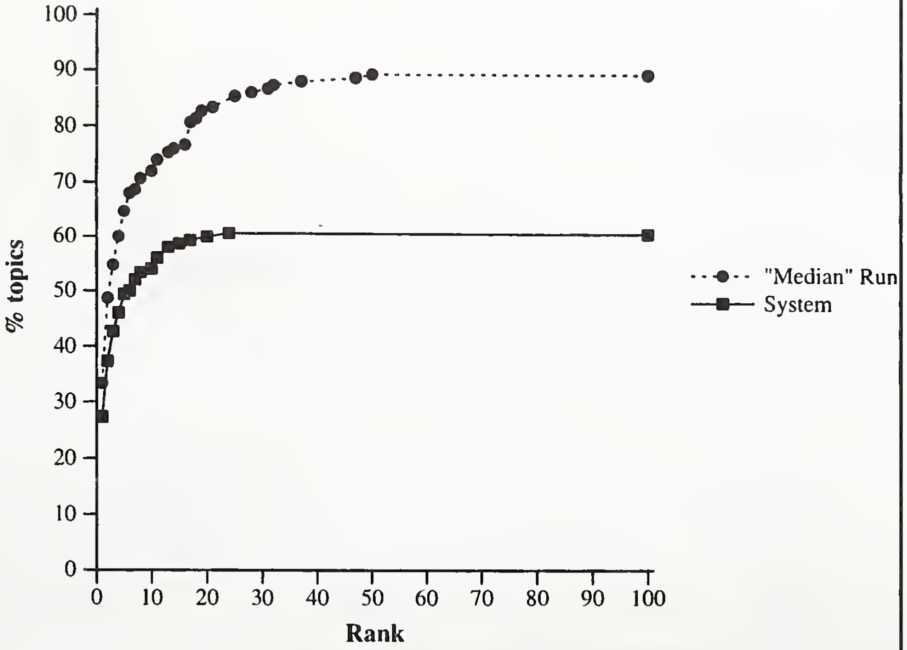
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	UAmsT02WnTm
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.260
Num found at rank 1	30 (20.0%)
Num found in top 10	58 (38.7%)
Num not found in top 100	83 (55.3%)



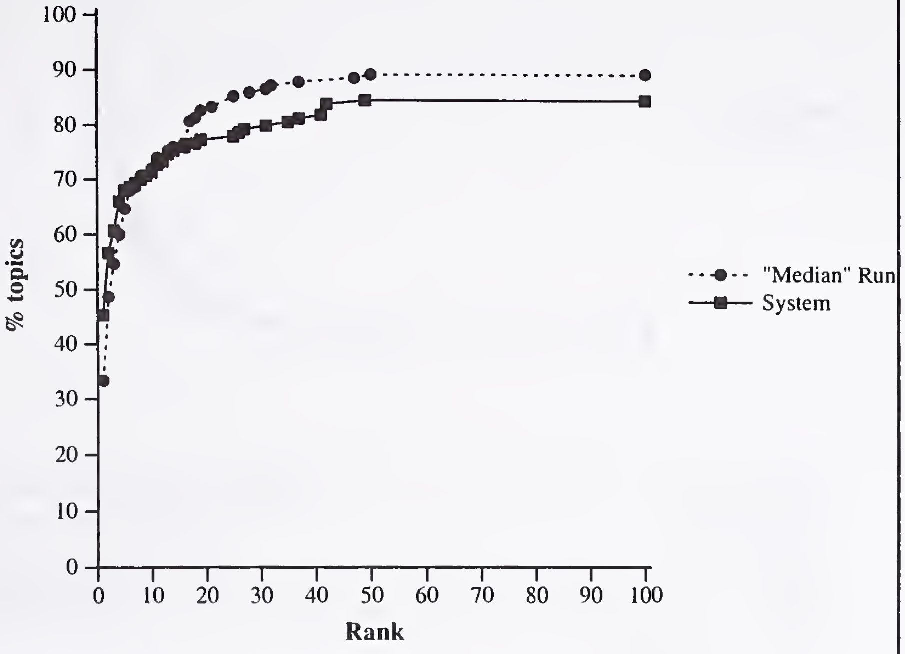
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	UAmsT02WnTmA
Run Description	NODOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.367
Num found at rank 1	41 (27.3%)
Num found in top 10	81 (54.0%)
Num not found in top 100	59 (39.3%)



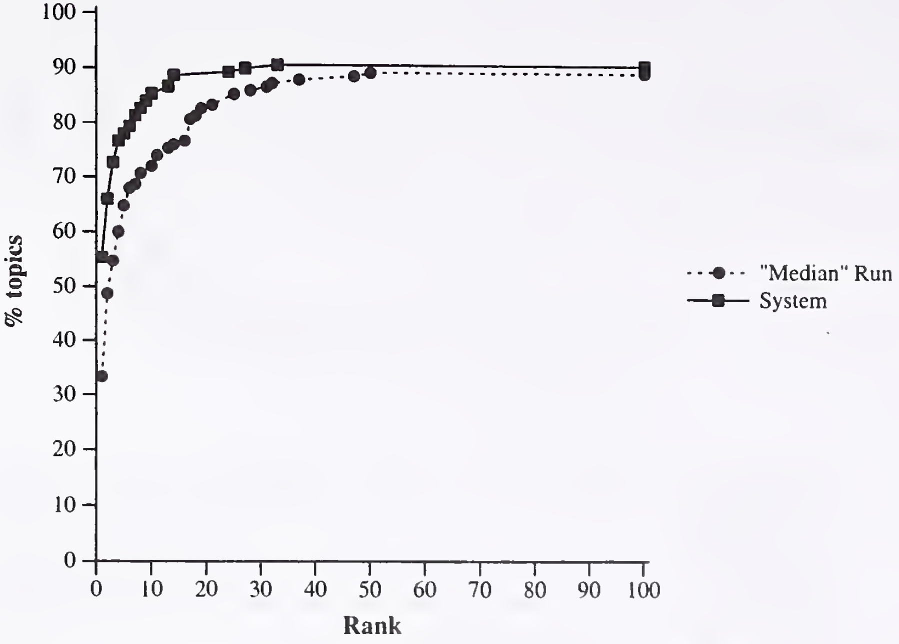
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	uog06c
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.552
Num found at rank 1	68 (45.3%)
Num found in top 10	107 (71.3%)
Num not found in top 100	23 (15.3%)



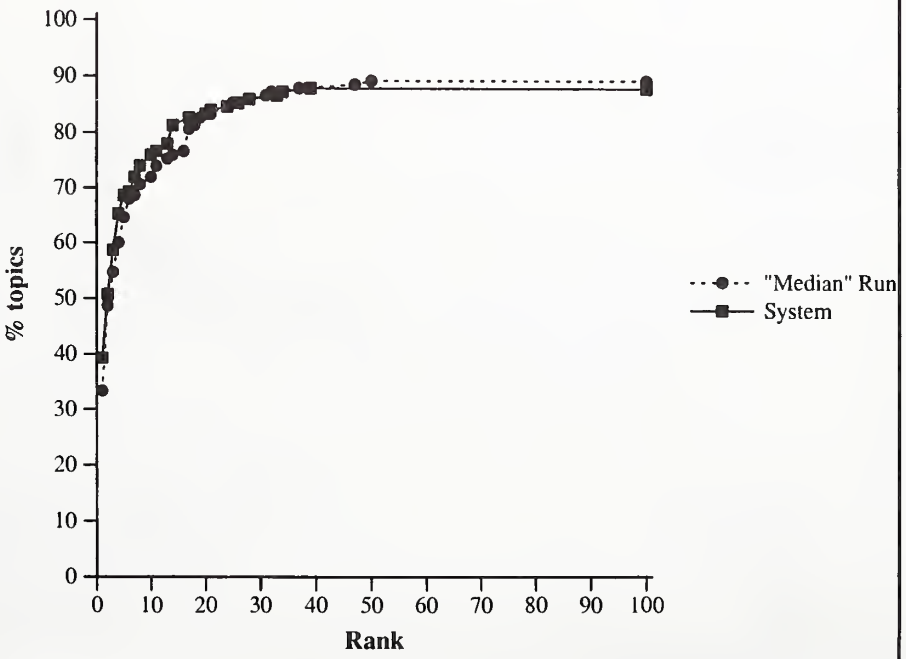
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	uog07cta
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.654
Num found at rank 1	83 (55.3%)
Num found in top 10	128 (85.3%)
Num not found in top 100	14 (9.3%)



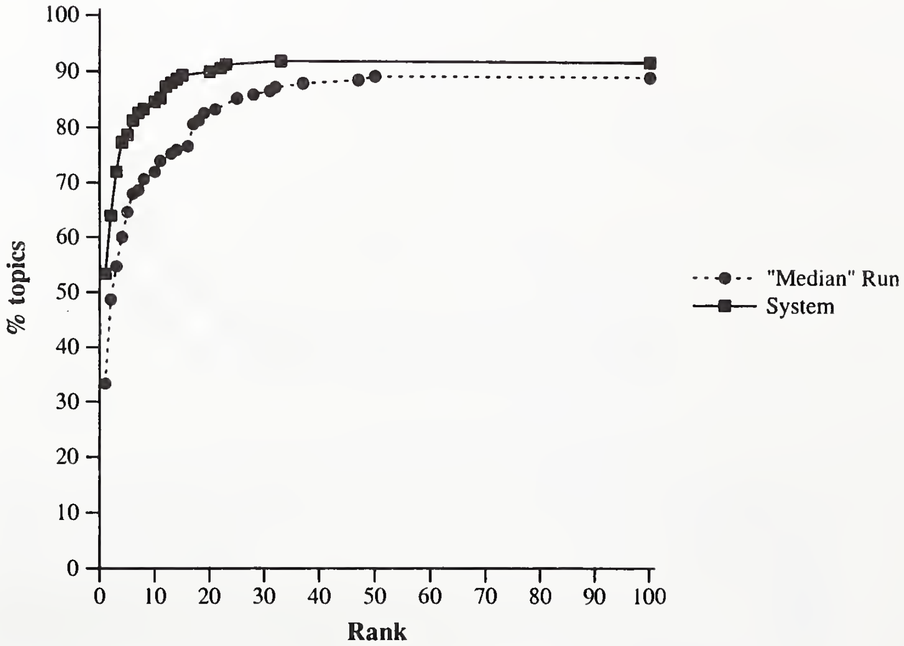
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	uog08ctap
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.516
Num found at rank 1	59 (39.3%)
Num found in top 10	114 (76.0%)
Num not found in top 100	18 (12.0%)



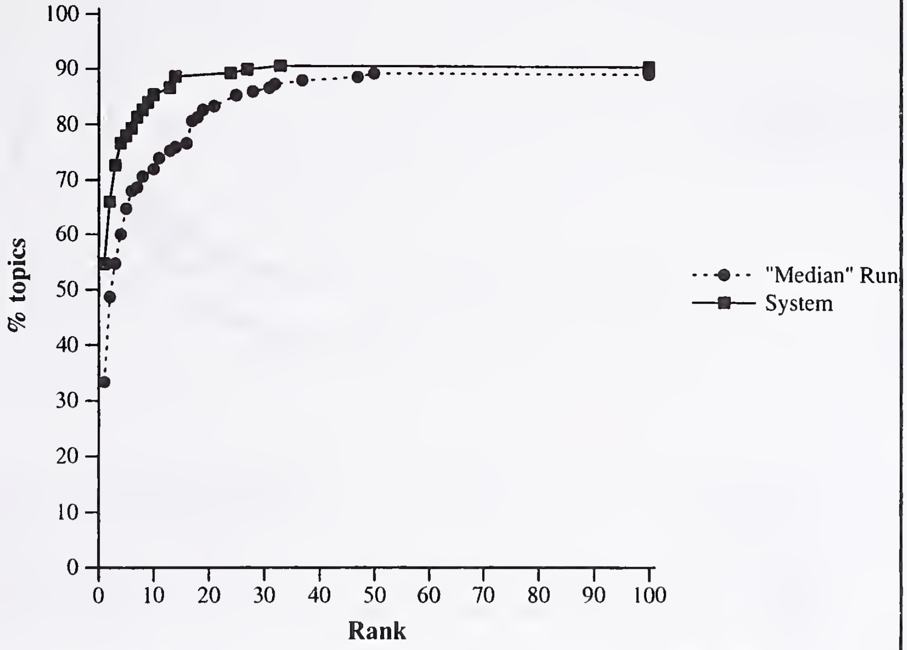
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	uog09cta2
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.643
Num found at rank 1	80 (53.3%)
Num found in top 10	127 (84.7%)
Num not found in top 100	12 (8.0%)



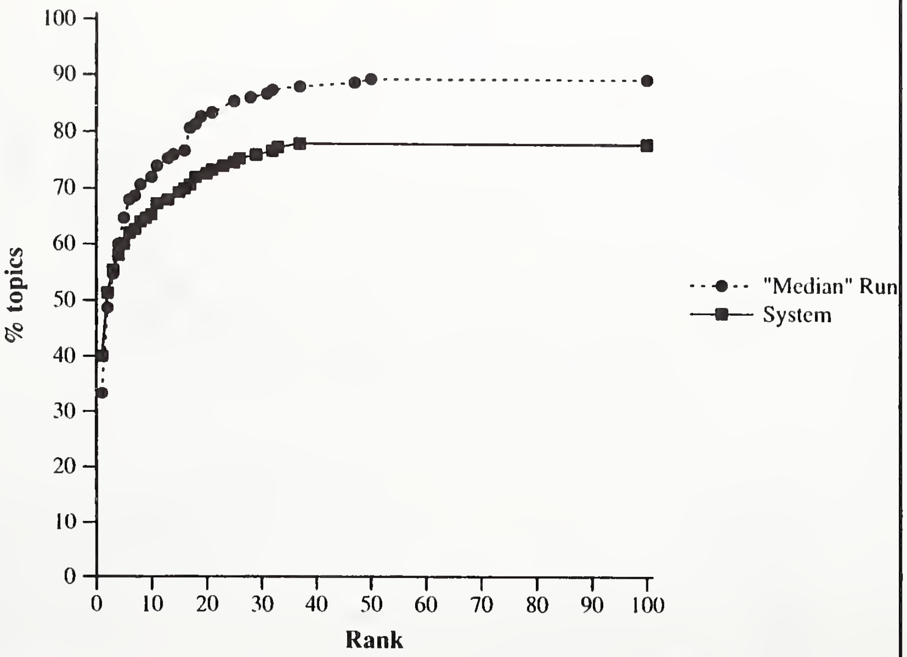
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	uog10ctad
Run Description	DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Num topics	150
Mean reciprocal rank	0.651
Num found at rank 1	82 (54.7%)
Num found in top 10	128 (85.3%)
Num not found in top 100	14 (9.3%)



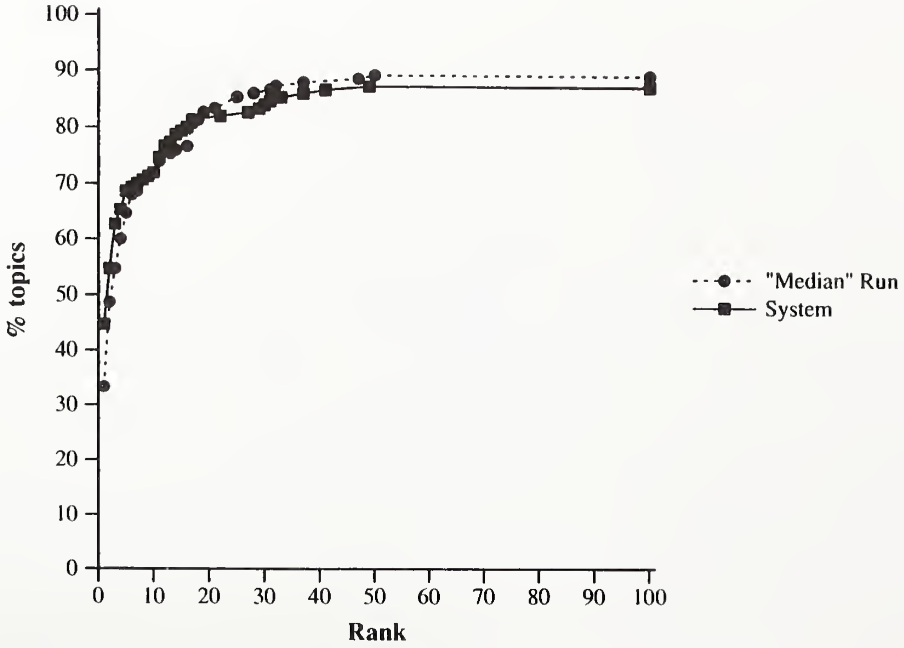
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	uicnp01
Run Description	NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Num topics	150
Mean reciprocal rank	0.495
Num found at rank 1	60 (40.0%)
Num found in top 10	98 (65.3%)
Num not found in top 100	33 (22.0%)



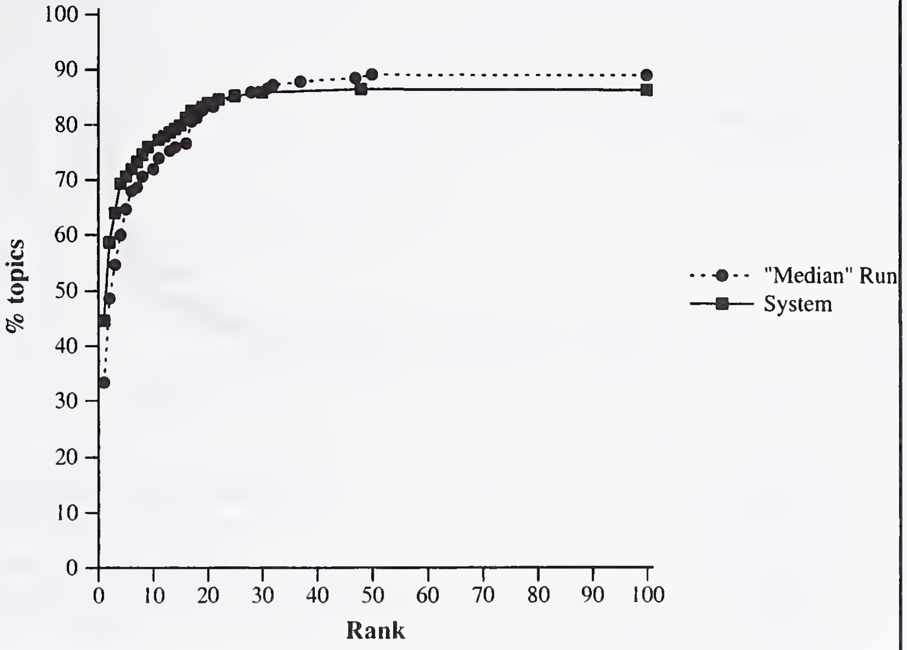
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	uicnp02
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.550
Num found at rank 1	67 (44.7%)
Num found in top 10	108 (72.0%)
Num not found in top 100	19 (12.7%)



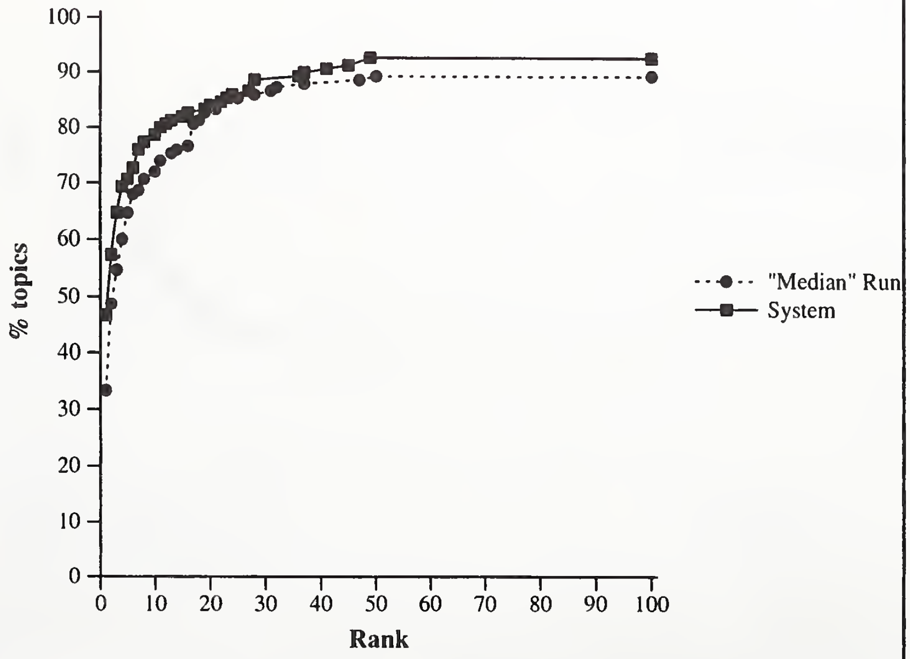
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	uicnp03
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.564
Num found at rank 1	67 (44.7%)
Num found in top 10	114 (76.0%)
Num not found in top 100	20 (13.3%)



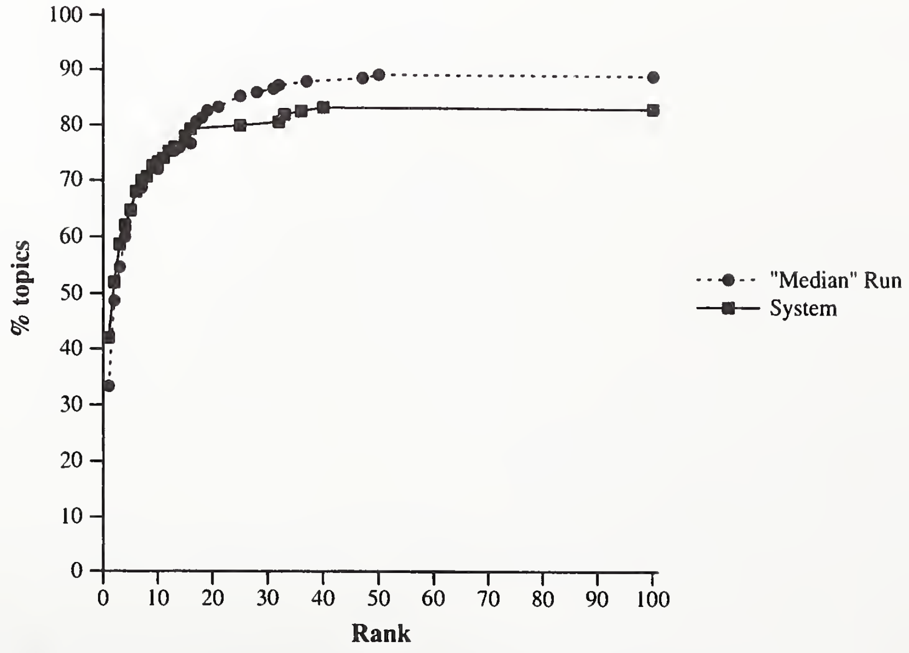
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	MU106
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.576
Num found at rank 1	70 (46.7%)
Num found in top 10	118 (78.7%)
Num not found in top 100	11 (7.3%)



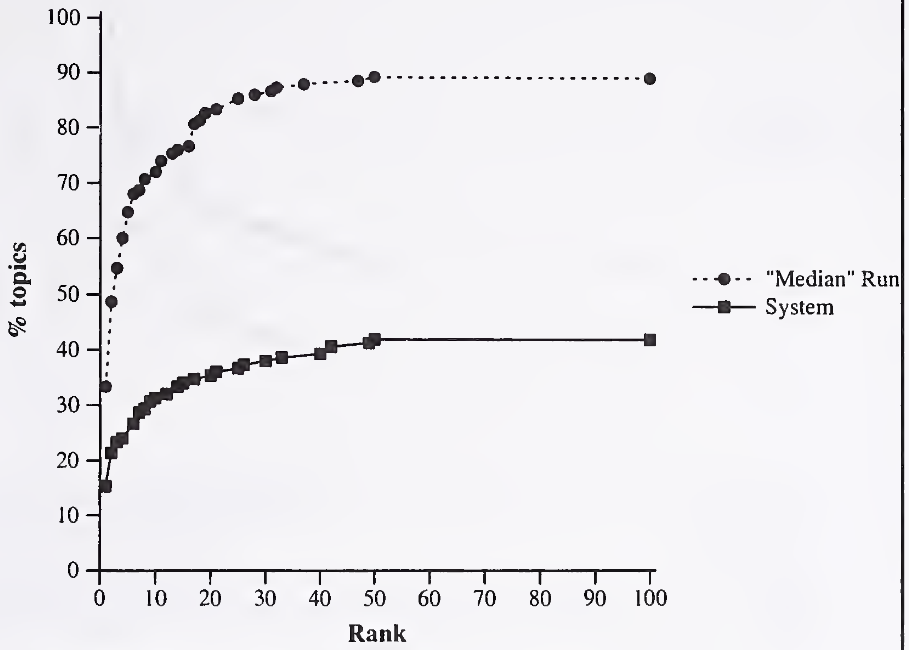
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	MU208
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.524
Num found at rank 1	63 (42.0%)
Num found in top 10	110 (73.3%)
Num not found in top 100	25 (16.7%)



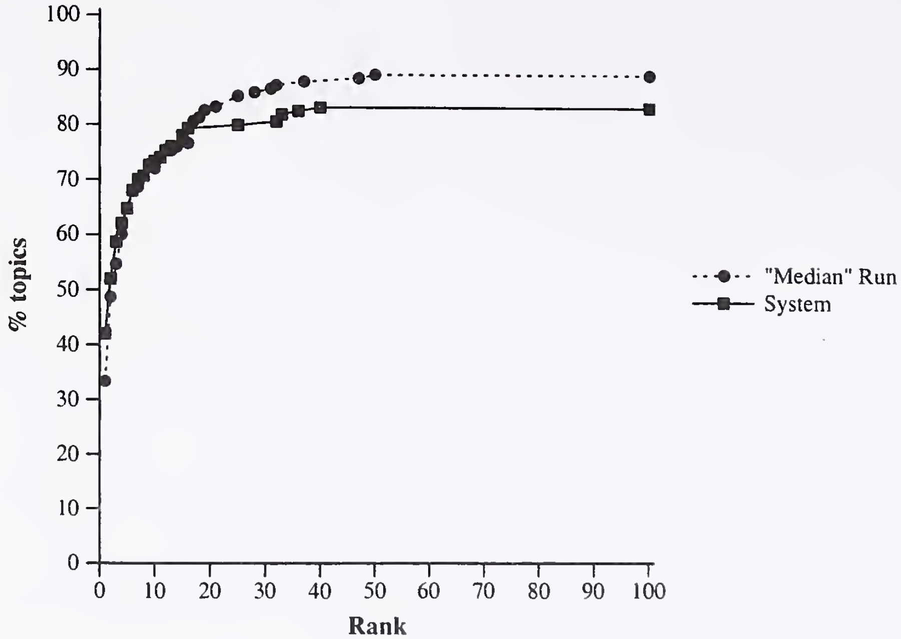
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	MU307
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.207
Num found at rank 1	23 (15.3%)
Num found in top 10	47 (31.3%)
Num not found in top 100	87 (58.0%)



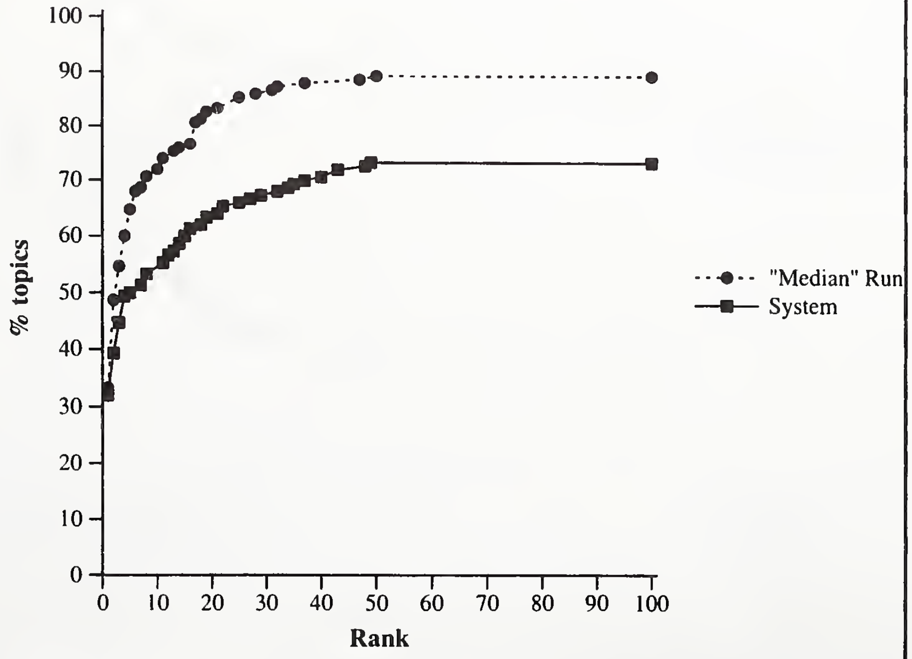
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	MU609
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.524
Num found at rank 1	63 (42.0%)
Num found in top 10	110 (73.3%)
Num not found in top 100	25 (16.7%)



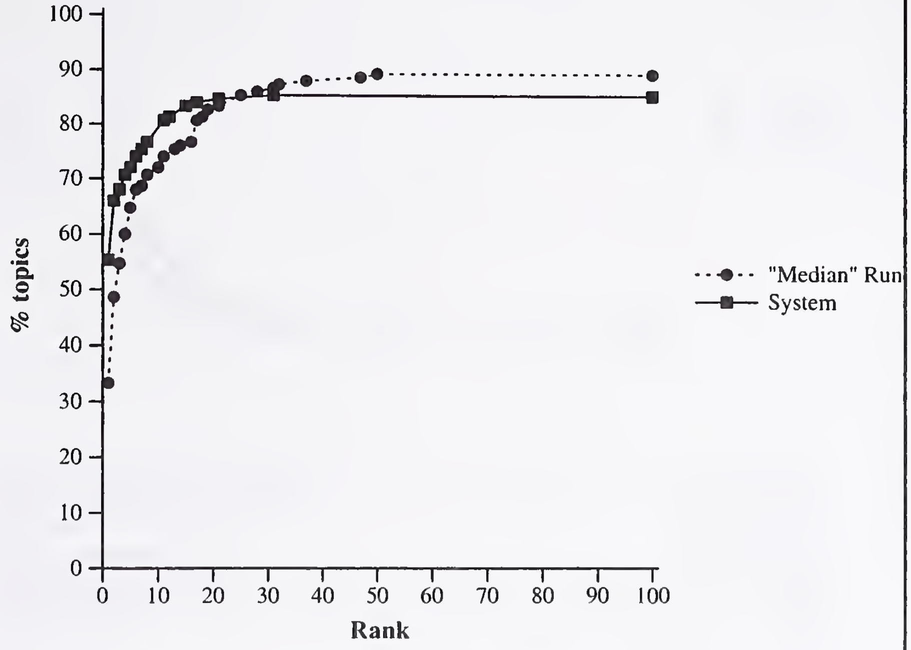
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	MU80A
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.402
Num found at rank 1	48 (32.0%)
Num found in top 10	80 (53.3%)
Num not found in top 100	40 (26.7%)



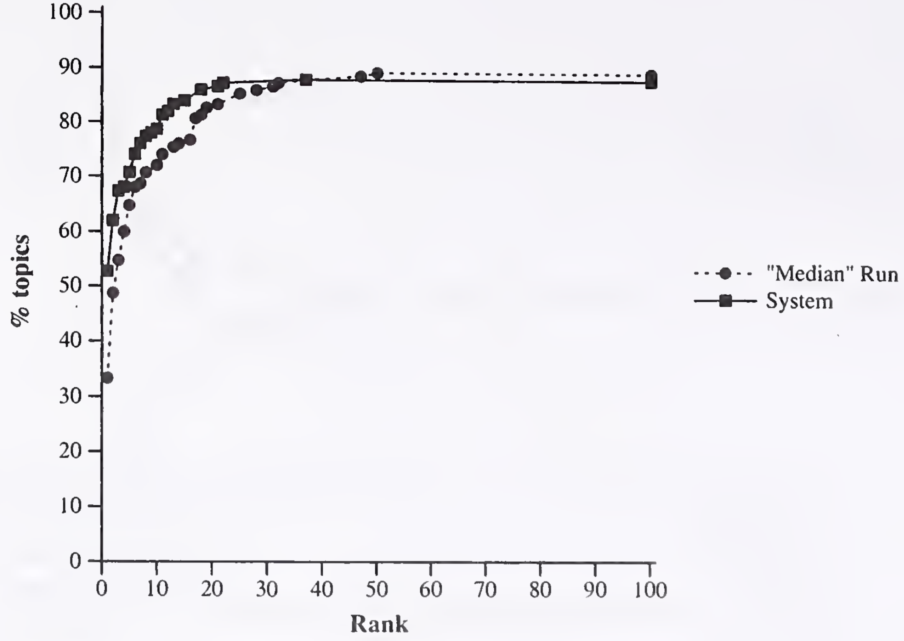
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	UniNEnp1
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.636
Num found at rank 1	83 (55.3%)
Num found in top 10	115 (76.7%)
Num not found in top 100	22 (14.7%)



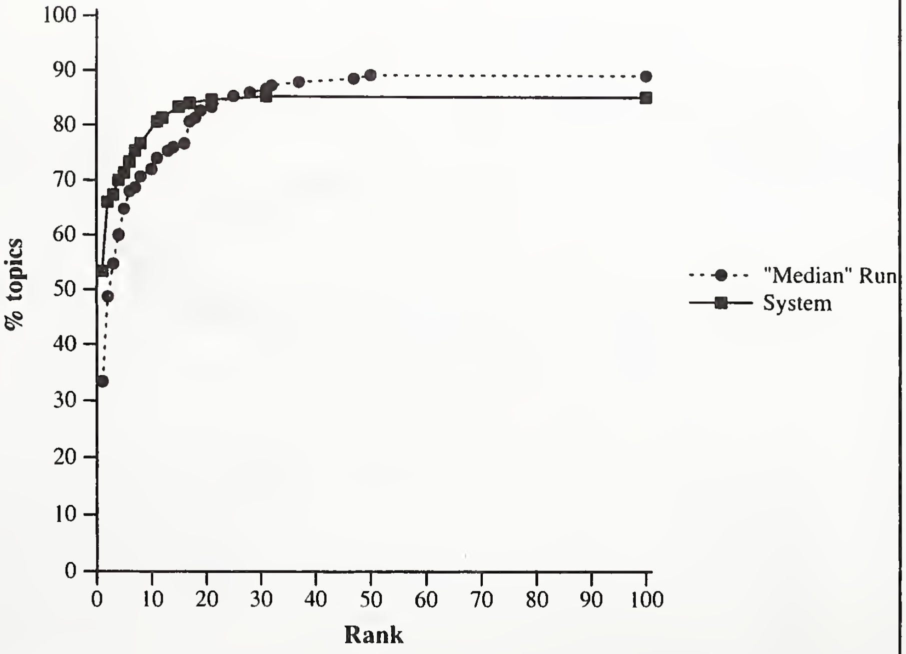
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	UniNEnp2
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.616
Num found at rank 1	79 (52.7%)
Num found in top 10	118 (78.7%)
Num not found in top 100	18 (12.0%)



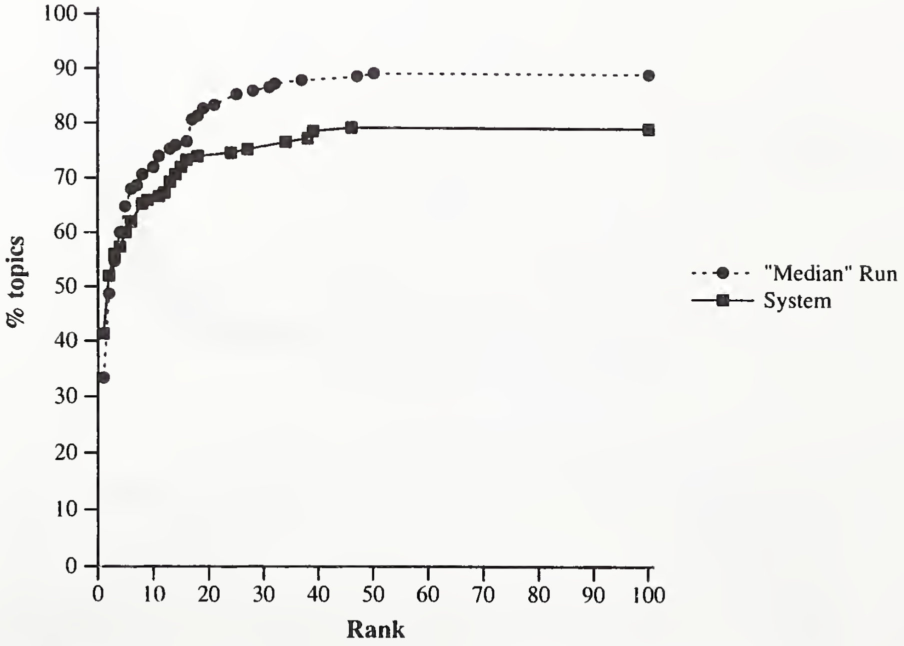
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	UniNEnp3
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.625
Num found at rank 1	80 (53.3%)
Num found in top 10	115 (76.7%)
Num not found in top 100	22 (14.7%)



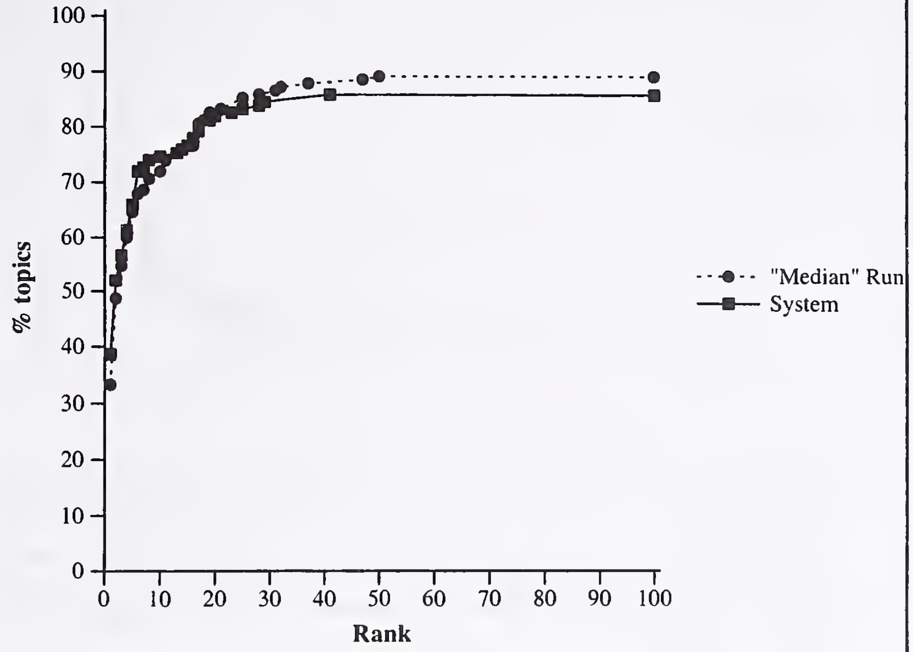
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	UniNEnp4
Run Description	DOCSTRUCT, ANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.504
Num found at rank 1	62 (41.3%)
Num found in top 10	99 (66.0%)
Num not found in top 100	31 (20.7%)



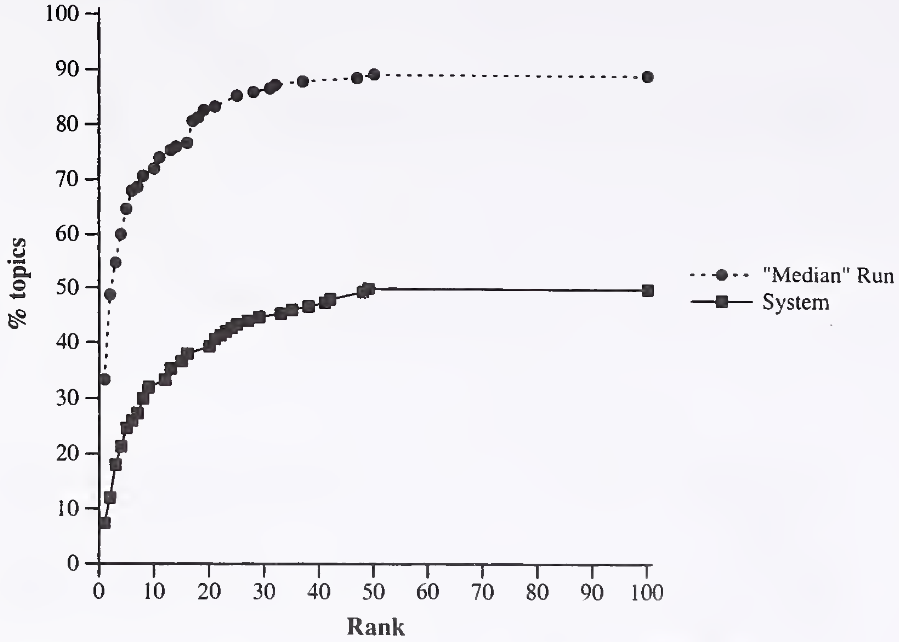
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	uwmtBW0
Run Description	NODOCSTRUCT, NOANCHORTEXT, NOLINKSTRUCT
Num topics	150
Mean reciprocal rank	0.509
Num found at rank 1	58 (38.7%)
Num found in top 10	112 (74.7%)
Num not found in top 100	21 (14.0%)



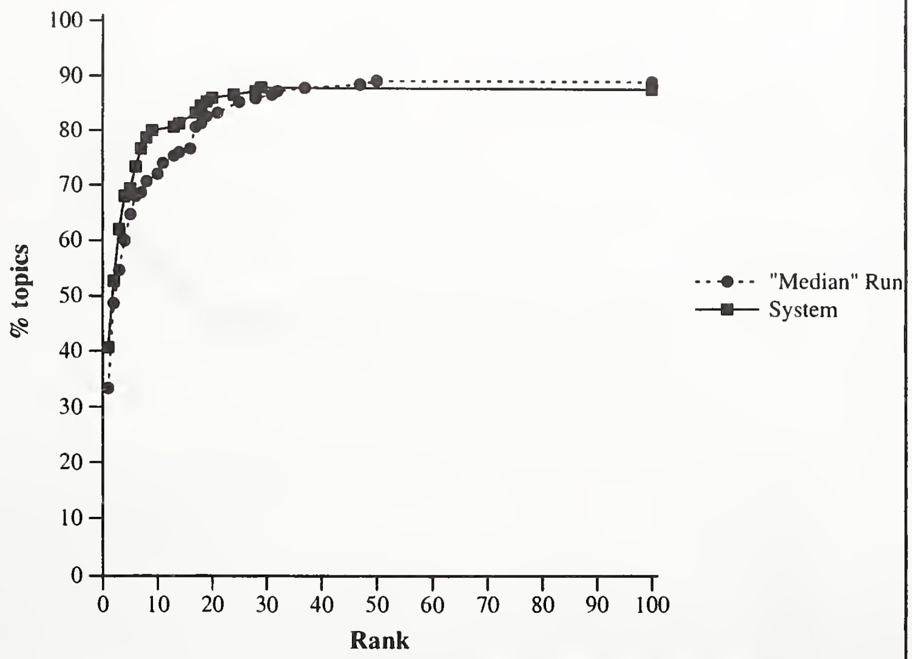
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	uwmtBW1
Run Description	NODOCSTRUCT, NOANCHORTEXT, LINKSTRUCT
Num topics	150
Mean reciprocal rank	0.150
Num found at rank 1	11 (7.3%)
Num found in top 10	48 (32.0%)
Num not found in top 100	75 (50.0%)



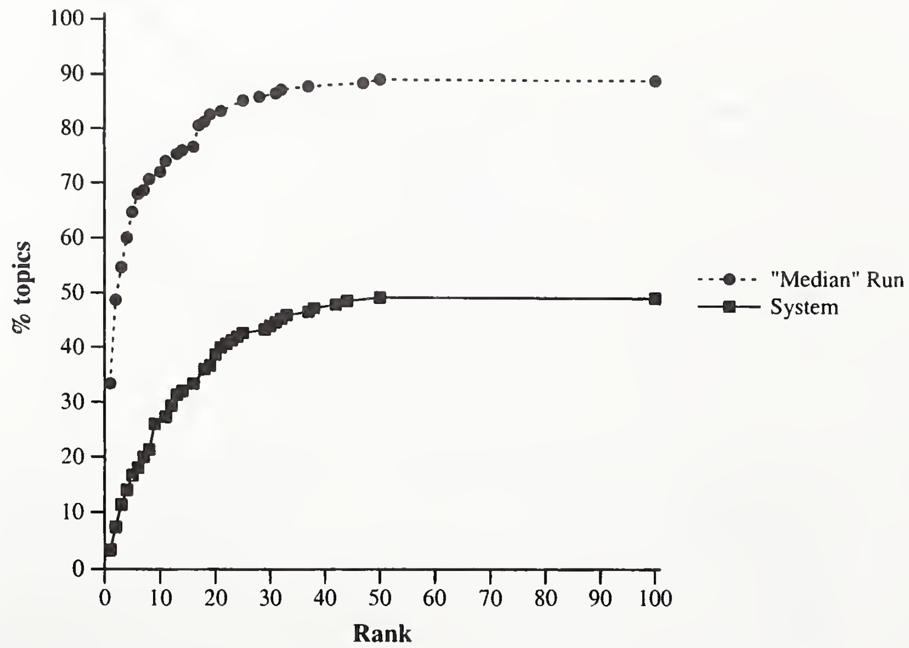
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	uwmtBW2
Run Description	NODOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Num topics	150
Mean reciprocal rank	0.535
Num found at rank 1	61 (40.7%)
Num found in top 10	120 (80.0%)
Num not found in top 100	18 (12.0%)



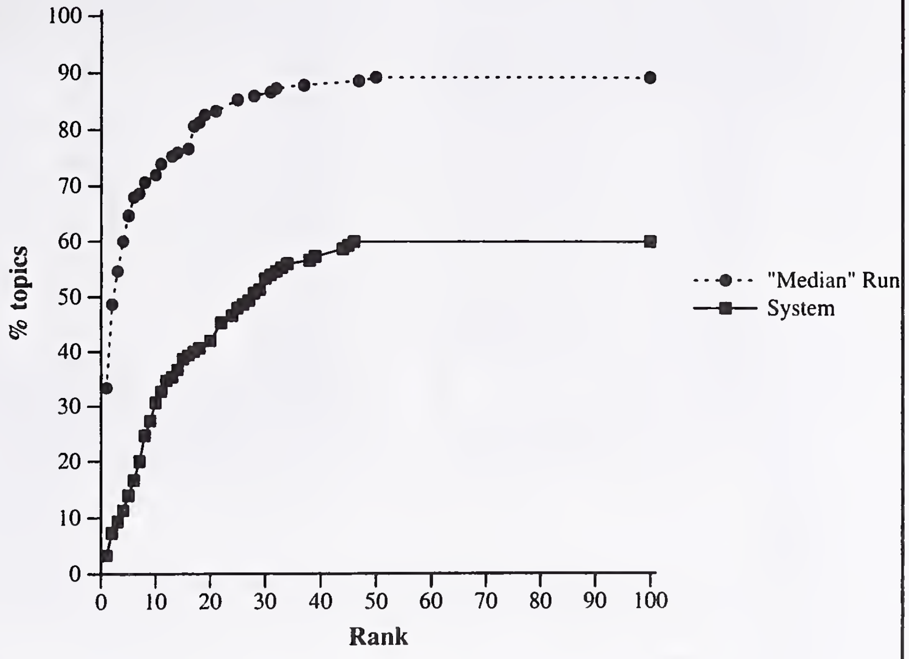
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	uwmtBW3
Run Description	NODOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Num topics	150
Mean reciprocal rank	0.103
Num found at rank 1	5 (3.3%)
Num found in top 10	39 (26.0%)
Num not found in top 100	76 (50.7%)



Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	uwmtBW4
Run Description	NODOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Num topics	150
Mean reciprocal rank	0.106
Num found at rank 1	5 (3.3%)
Num found in top 10	46 (30.7%)
Num not found in top 100	60 (40.0%)



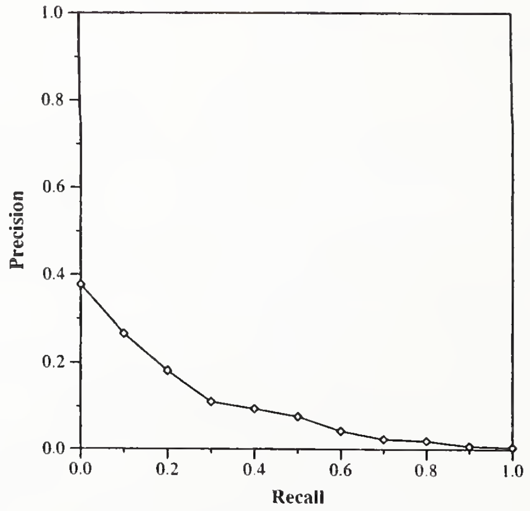
Cumulative % of topics that retrieve named page by given rank

Summary Statistics	
Run ID	yedi01no
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics	49

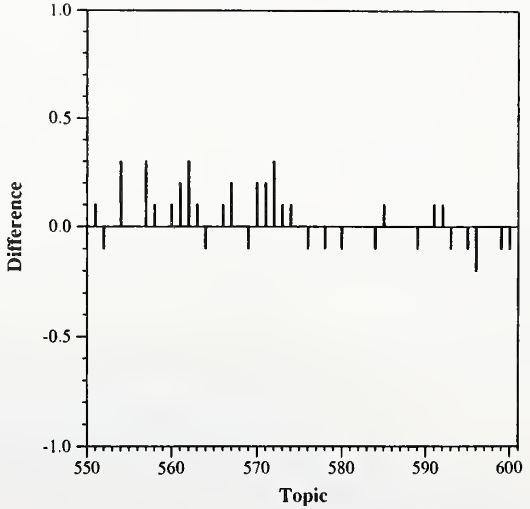
Total number of documents over all topics	
Retrieved:	31072
Relevant:	1574
Rel-ret:	652

Recall Level Precision Averages	
Recall	Precision
0.00	0.3770
0.10	0.2657
0.20	0.1823
0.30	0.1114
0.40	0.0952
0.50	0.0772
0.60	0.0437
0.70	0.0245
0.80	0.0204
0.90	0.0076
1.00	0.0047
Average precision over all relevant docs	
non-interpolated	0.0942

Document Level Averages	
	Precision
At 5 docs	0.1796
At 10 docs	0.1469
At 15 docs	0.1224
At 20 docs	0.1235
At 30 docs	0.1027
At 100 docs	0.0600
At 200 docs	0.0411
At 500 docs	0.0216
At 1000 docs	0.0133
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1193



Recall-Precision Curve



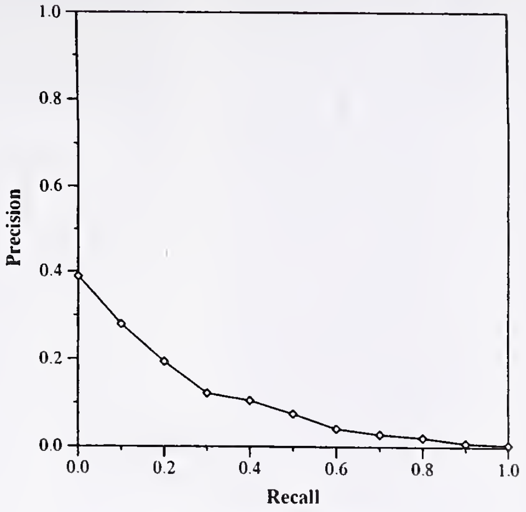
Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID	yedi01
Run Description	Realistic; DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Number of Topics	49

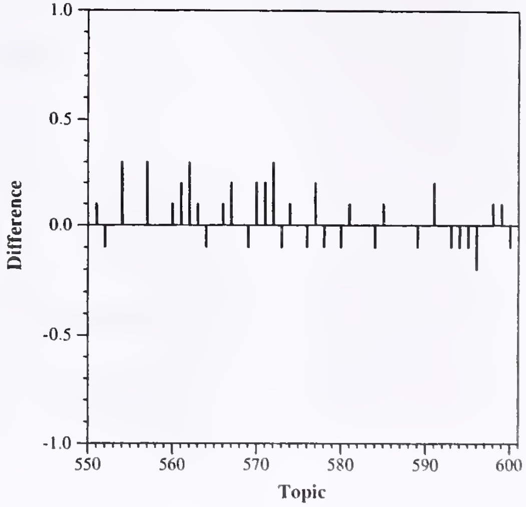
Total number of documents over all topics	
Retrieved:	31072
Relevant:	1574
Rel-ret:	640

Recall Level Precision Averages	
Recall	Precision
0.00	0.3886
0.10	0.2797
0.20	0.1945
0.30	0.1223
0.40	0.1060
0.50	0.0753
0.60	0.0411
0.70	0.0282
0.80	0.0207
0.90	0.0081
1.00	0.0045
Average precision over all relevant docs	
non-interpolated	0.0986

Document Level Averages	
	Precision
At 5 docs	0.1755
At 10 docs	0.1510
At 15 docs	0.1361
At 20 docs	0.1255
At 30 docs	0.1136
At 100 docs	0.0639
At 200 docs	0.0424
At 500 docs	0.0215
At 1000 docs	0.0131
R-Precision (precision after R docs retrieved (where R is the number of relevant documents))	
Exact	0.1298

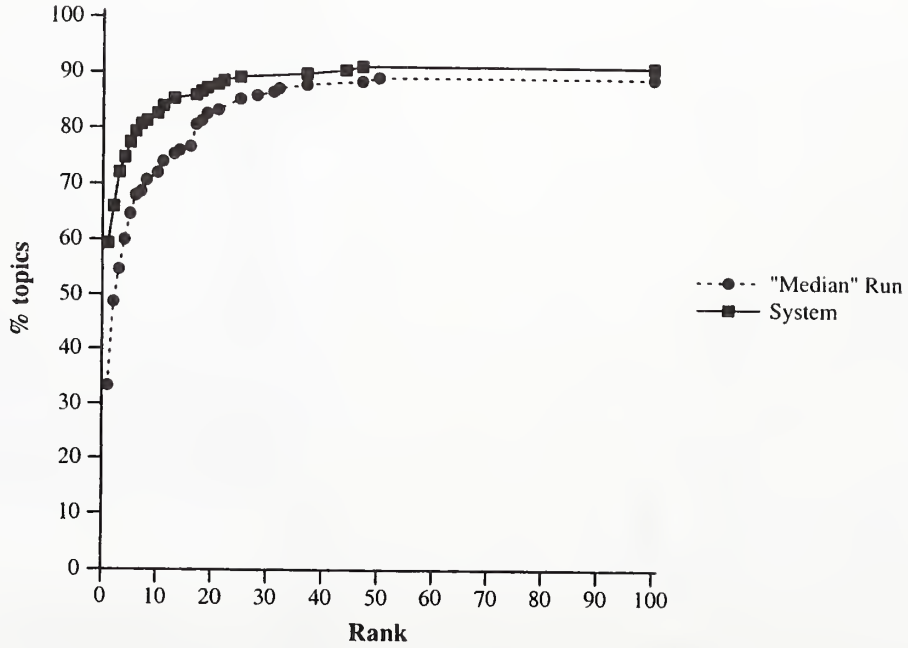


Recall-Precision Curve



Difference from Median in Precision at 10 per Topic

Summary Statistics	
Run ID	yenp01
Run Description	DOCSTRUCT, ANCHORTEXT, LINKSTRUCT
Num topics	150
Mean reciprocal rank	0.671
Num found at rank 1	89 (59.3%)
Num found in top 10	124 (82.7%)
Num not found in top 100	13 (8.7%)



Cumulative % of topics that retrieve named page by given rank

☆U.S. GOVERNMENT PRINTING OFFICE: 2003/496-169/86303

NIST Technical Publications

Periodical

Journal of Research of the National Institute of Standards and Technology—Reports NIST research and development in those disciplines of the physical and engineering sciences in which the Institute is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Institute's technical and scientific programs. Issued six times a year.

Nonperiodicals

Monographs—Major contributions to the technical literature on various subjects related to the Institute's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NIST, NIST annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NIST under the authority of the National Standard Data Act (Public Law 90-396). NOTE: The Journal of Physical and Chemical Reference Data (JPCRD) is published bimonthly for NIST by the American Institute of Physics (AIP). Subscription orders and renewals are available from AIP, P.O. Box 503284, St. Louis, MO 63150-3284.

Building Science Series—Disseminates technical information developed at the Institute on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NIST under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NIST administers this program in support of the efforts of private-sector standardizing organizations.

Order the **following** *NIST publications—FIPS and NISTIRs—from the National Technical Information Service, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NIST pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NIST Interagency or Internal Reports (NISTIR)—The series includes interim or final reports on work performed by NIST for outside sponsors (both government and nongovernment). In general, initial distribution is handled by the sponsor; public distribution is handled by sales through the National Technical Information Service, Springfield, VA 22161, in hard copy, electronic media, or microfiche form. NISTIR's may also report results of NIST projects of transitory or limited interest, including those that will be published subsequently in more comprehensive form.

U.S. Department of Commerce
National Institute of Standards and Technology
Gaithersburg, MD 20899-0001

Official Business
Penalty for Private Use $300